Facilitating Collaborative Ontology-Based Annotations in Communities of Interest

vom Fachbereich Informatik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation von

Dipl.-Inf. Malte Kiesel

Datum der Aussprache: 16.05.2014

Gutachter

Prof. Dr. Andreas Dengel, TU Kaiserslautern
Prof. Dr. Randy Goebel, University of Alberta
Prof. Dr. Theo Härder, TU Kaiserslautern

Vorsitzender der Prüfungskommission

Prof. Dr. Stefan Deßloch, TU Kaiserslautern

Dekan

Prof. Dr. Klaus Schneider, TU Kaiserslautern

D 386

Acknowledgements

This thesis is the result of about eight years of work during which many people were involved directly and indirectly. Additionally to the co-workers and students mentioned in detail in Section 1.5, I would like to express some special thanks.

Thanks go to Professor Andreas Dengel for his feedback on my work and, in general, heading a department that allowed carrying out this research. Further I want to thank Professor Randy Goebel for his feedback and ideas for the thesis and Professor Theo Härder for his help on short notice.

Also, I would like to thank Michael Sintek und Ludger van Elst, who mentored me during my first days in research, and Sven Schwarz, Gunnar Grimnes, and Leo Sauermann, who provided key influences in research as well as good company during time off. My thanks go to Marcus Eichenberger-Liwicki for his detailed feedback on my thesis. Special thanks go to the people who participated in the evaluation of the systems, my friends (who provided the necessary nudges to get the work done), and my family.

Malte Kiesel

Abstract

If an automated system is tasked to provide services such as search or clustering of information on an information repository, the quality of the output depends a lot on the information that is available to the system in machine-readable form. Simple text, for example, is machine-readable only in a very limited sense. Advanced services typically need to derive other representations of the text (e.g., sets of keywords) as input for their core algorithms. Some services might need information that cannot be derived from the resource in question alone, but is available as separate metadata only, such as usage information. *Annotations* can be used to carry this information.

This thesis focuses on so-called *ontology-based annotations*. In contrast to other forms of annotations such as tags (arbitrary strings that users can assign to resources), ontology-based annotations conform to a predefined data structure and class hierarchy. An advantage of this approach is that rich information can be stored in a well-structured way in the annotations; a drawback is that users need to be familiar with the hierarchy and other design decisions of the underlying ontology used for annotations.

Two scenarios are considered in this thesis: First, a document-based scenario in which text annotations are used to represent both information about the text content and usage and user context information in a multi-user setting with mostly objective annotation criteria; second, a resource-based scenario whose annotation model focuses on multi-user settings with subjective annotation criteria, using (dis-)similarities in user annotations to derive user similarity metrics, and building personalized views from this information.

Finally, the prototypical systems that have been developed throughout this thesis get evaluated, proving the concepts presented in this thesis.

Contents

I

Introduction

CHAPTER 1
Introduction

The results presented in this thesis are the outcome of several years of work that focused on how to capture, represent, and use information occurring when working with multi-user knowledge repositories. The types and nature of this information are manifold, ranging from explicit representations of the knowledge contained in resources via user attention information to user ratings concerning individual resources available in the repositories. Due to this wide variety, finding a common representation of the information to be captured is a challenge. In this thesis, the approach of representing this information in the form of *annotations* is pursued. Once the task of pure capturing is solved, the next question is how to handle the aggregated knowledge of the multiple users that use the repositories. Especially for types of information that have a subjective nature, people will often not agree fully. In this thesis, the solution approach pursued is weighting statements of different users differently to build a *personalized view* of the data available in the repository.

1.1. Motivation

In current knowledge repositories, navigation by means of search and browsing is increasingly becoming a challenge. Finding items or text passages of interest in a large pool of information is often tedious, even with technologies such as full text search. At the same time, the repository is mostly opaque to the machine storing it; it really is a *data* repository only, allowing merely syntactic access. Advanced features such as intelligent aggregation of information, personalized digests, weighting of information by personalized importance, or finding inconsistencies in the information available, is not possible, or limited to approximations mostly working on syntactic level.

Only Syntactic Access to Knowledge Repositories

With the advent of Web 2.0, tagging information items (i.e., assigning descriptive strings to items) is the predominant way of providing structure in the information pool. Tags can be used in search and browsing—one well-known method for explorative browsing using Tags is the *Tag Cloud*.

Tagging

Weak Semantics

However, time has shown that tagging is a partial solution only. Tags introduce ambiguity, are not necessarily used in a coherent way by different users, and are lacking clear semantics: The semantics of the relationship of a Tag to the item it is assigned to is typically not made explicit, and semantics of the Tag itself are unclear due to the fact that a Tag is, to a computer, just a string without any additional information. Most of these shortcomings are a result of the deliberate restriction of tagging to simple concepts and usage.

Ontology-Based Annotations

Another approach of implementing annotations is to use ontology-based annotations. Ontologies are typically defined by ontology engineers based on input from domain experts. Since the structure and semantics of an ontology are well defined, this lays the foundations for inferring new facts from existing annotations, and providing services with higher accuracy and less requirements on annotation volume than would otherwise be necessary.

Explicit Semantics

The explicit semantics of ontology-based annotations allow discovering inconsistencies in annotations. In annotation systems that lack *a priori semantics*, this is not possible, or can only be done with post-processing that assigns semantics to Tags *a posteriori*, which is potentially error prone. A special case of inconsistencies in annotations is inconsistencies created by different users in multi-user environments[1]. If creation of inconsistent annotations is not blocked at data entry time, such annotations represent *subjective annotations*, and a set of inconsistent annotations represent an *opinion clash*. Systems lacking semantic background knowledge cannot easily detect such opinion clashes. Semantically enabled system can, and they can even use differing opinions to derive advanced concepts like competence and trust metrics between users: If two users agree concerning their annotations (i.e., there are no opinion clashes in the semantics of their annotations; they have a high *user similarity* concerning their statements), it is very likely they both (i) trust each other, and (ii) share the same opinions. Aggregating annotations (i.e., building one concise description of a resource by summarizing all its annotations by meaning rather than provenance) benefits from user similarity metrics as well: When aggregating individual annotations, user similarity can be used for weighting, resulting in completely personalized views of the data available in the system.

Subjective Annotations

Trust

Personalized Views

Contextualized Annotations

Another benefit of ontology-based annotations is that they can carry further metadata (i.e., link to further items of interest), and can get enriched using additional information such as context information (i.e., what projects was the user working on when creating the annotation). This additional information in annotations can get used for advanced search and building personalized views.

[1] The set of annotations of just one user can also be inconsistent. However, the implications of this are very different from the multi-user case, which this thesis focuses on.

Steep Learning Curve for Ontology-Based Systems

A challenge in ontology-based annotation systems is the fact that most systems of this kind require thorough knowledge of the ontologies used and, thus, have a steep learning curve. Also, the process of annotating information items is work intensive: whenever an annotation is to be created, the user has to find the appropriate resource in the ontology, reference it, and possibly include additional information.

Automatically Creating Annotations

Ways of helping with creating ontology-based annotations are required. One promising approach is tapping ubiquitous sources of information during use, and generating annotations based on these. In special circumstances, this is even possible automatically. Technology such as eye tracking and incorporating user context information into the annotating process is key to this approach. However, it is limited to certain kinds of annotations. Thus, the main goal must be enabling the user to create lots of annotations quickly, with high accuracy, and with minimal effort. Annotation recommendation—showing the user potential annotations and letting him choose and fine-tune them—is central to this approach. To motivate the user during annotation, *instant gratification* by maximizing the gain drawn from user annotations is desirable. Depending on the functionality that is using annotations, this can take several forms: for example, in case the goal is to improve quality of item recommendations (i.e., rank items according to the assumed importance to the user), the annotation recommender can request most discriminating statements about items first.

Annotation Recommenders

Two Scenarios

Two scenarios highlighting different requirements are used throughout this thesis. The focus of the first scenario are text document repositories and functionality based on fine-grained ontology-based text document annotations. The second scenario is about resource-level ontology-based annotations. In the following, the scenarios get explained in more detail.

Document-Based Scenario

Large Corpora and Long Documents

In the document-based scenario, the subject of interest is large corpora of (possibly long) documents in an intranet setting with knowledge workers numbering in the low two-digit range. Knowledge workers read these documents, annotate them with personal notes, want to be able to search for key concepts, discover links between documents, and re-find things previously read.

Normal document repositories are typically implemented using text documents in filesystems or network/Web-based storage, possibly with additional functionality such as full-text search or tagging support. These systems have trouble fulfilling the requirements outlined above:

- a lot of potentially valuable usage information by the user (attention and context information) does not get captured
- annotating specific document parts is not possible
- semantic links between documents cannot be discovered automatically

Solution Approach

The solution approach presented in this thesis builds on fine-grained annotations of (parts of) text documents, user support in creating these annotations, support for automatically generated annotations, and annotation-based advanced search and browsing. The implementation outlined in this thesis is based on a Semantic Wiki that allows annotating text parts stored in it with arbitrary information based on ontologies.

Resource-Based Scenario

Resource Descriptions

In the resource-based scenario, the subject of interest is a repository of (descriptions of) resources such as books, software, or Web links. The repository is used by multiple users, both reading and adding to it. Concerning search, browsing, and item recommendation needs, other approaches to this scenario such as Flickr.com employ tagging. However, this has some drawbacks:

- different people use different meanings for the same Tags
- with different people employing different sets of Tags, Recall in search is influenced negatively
- without means to detect inconsistencies, resource and expert recommendation is limited

Solution Approach

The approach pursued in this thesis to address these shortcomings includes allowing users to annotate items with subjective statements. These statements get created based on annotation ontologies. Along with the statements, users can specify confidence and the degree of truth associated with the statements. This allows users to express uncertainty concerning their statements as well as to express negation. Handling and keeping provenance of annotations allows drawing inferences concerning trust between users. Using this trust, the system can build personalized views, in which individual statements get aggregated by weighting them with their associated trust. With the fine-grained aggregated information available to the system, enhanced recommender functionality such as expert recommendation becomes possible.

Both scenarios and approaches can get combined and, in fact, are combined for the evaluation carried out in this thesis. However, the focus of both scenarios differs.

Key Differences of the Scenarios

In the first scenario, user context and user attention information is of special importance, as tapping these information sources allows annotation of long documents with little user effort and large gains for search and navigation. In the second scenario, user context information is of far less importance, and user attention information in the form available in the first scenario (paragraph-level attention information) is not available at all. Also, intra-document annotations that possibly refer each other (*structural annotations*) occur only in the first scenario. In general, the types of metadata handled in the first scenario are broader than the types that are of main concern in the second scenario.

First Scenario

The second scenario focuses on other aspects. There, annotations have resource-level granularity, which allows building user similarity metrics that would have sparsity issues in the first scenario[2]. Content-based annotations are the annotations that are most interesting in the second scenario. Additionally, the type of resources does not matter in this scenario: Be it text documents, books, board games, people – since the approach handles abstract notions of the resource in question, any type of resource can be handled. In the first scenario, supporting more types of resources would mean adding facilities to parse and annotate pictures, videos, music, etc., which is a huge task, and likely to result in a very complicated user interface.

Second Scenario

This concludes the overview over the key concepts and motivation of this thesis. In the following section, the aim of the research and hypotheses are presented.

1.2. Aim and Research Hypotheses

In the following, the main directions the research in this thesis is aiming at are listed. The research hypotheses then presented follow from these aims.

Enabling Semantics in Wikis

Wikis are an important tool in typical everyday work in communities of interest. Unfortunately, most of the data available in Wikis is strictly human-readable only; it is not available for automated processing, in a format that is machine-readable. Also, retrieval operations in Wikis are typically limited to rather syn-

[2]In order to detect conflicting user opinions in the first scenario, two users would have to annotate the *same text passage* (not only the same document) with conflicting annotations.

tactic retrieval operations (e.g., text search). *Semantic Wikis* aim at making the information contained in the Wiki available in a machine-readable format. One focus of this work is extending the Semantic Wiki approach to enable processing of automatically created annotations and capturing further metadata such as usage and user context information. At the same time, functionality to use this additional information in search and navigation is explored.

Facilitating Collaborative Semantic Annotations

Differing Opinions

Handling Inconsistencies

If many people work together in one semantically driven system, several challenges arise. One especially interesting challenge is how to deal with differences in opinions, and inconsistencies between statements of different users. In some cases, it is desirable just to disallow or highlight inconsistencies; however, for any statements or domains that are situated in a domain with high subjectivity, this is not an option. Instead, embracing the fact that user opinions differ, and using this to implement additional functionality is a viable approach.

Creating Semantic Annotation Recommenders

Helping Creating Annotations

Creating ontology-based annotations can be a tedious process as users need to know the ontologies or vocabulary used in the domain. Therefore, simplifying this is a major goal when creating ontology-based systems. One way to help the user is to proactively show possible annotations instead of requesting the user to enter annotation data (e.g., the annotation class). Several approaches for this are explored in this thesis.

Semantically Tapping Web-Based Services

Exposing Machine-Readable Data

Creating and maintaining the ontologies needed to implement ontology-based approaches is very work intensive. However, for many domains there already exist knowledge structures on the Web that wait to get tapped: for example, Wikipedia contains vast amounts of concepts and classifications of these concepts. Unfortunately, in the form it is available in Wikipedia, this information is not machine-readable. This thesis examines a wrapping approach to make additional data sources available in machine-readable form.

Research Hypotheses

Resource-Based Annotation Ontology

In the resource annotation scenario, the focus is describing resources by their features. In this thesis, a robust distributed model for annotating resources is investigated. Integral part of it is a class hierarchy for annotations, a class hierarchy for resources that can get described, and the ability to represent truth values and user confidence.

Hypothesis H1: The base ontology, implementing a feature hierarchy and supporting truth values and user confidence, allows intuitive expressive annotation of resources in a multi-user scenario.

Generating Semantic Annotations with Little Overhead

The scope of manual annotations is limited as creating annotations manually is tedious work and requires both vast domain knowledge and knowledge of the domain model used. Key to good adoption of ontology-based annotations is user support in creating these annotations; this lets the user learn about the domain while using the system, and speeds up the annotation process in general.

Hypothesis H2: Support when creating ontology-based annotations leads to higher annotation volume and, in general, better user acceptance.

Handling Inconsistent Annotations

If multiple users are contributing annotations, inconsistencies between annotations will arise. In traditional systems, annotations that contradict each other are undesirable and get typically blocked already at entry time. However, if the system handling annotations builds and updates a user similarity model based on analyzing contradicting annotations, additional functionality such as building personalized views becomes possible.

Hypothesis H3: Embracing inconsistencies and providing personalized views leads to improved user experience and better item recommendations.

Integration of External Ontology Sources

The process of annotating documents and other resources is not the only process that is work-intensive. Creating and maintaining ontologies that are used in the annotation process also comes at some cost. Extracting facts from Web resources

automatically is considered unreliable due to errors in the extraction process. However, by using domain rules and ontology restrictions, improvements can be implemented.

Hypothesis H4: Validation using domain rules results in high Precision in ontology and facts extraction.

1.3. Contributions

This thesis presents a number of contributions in the fields of Semantic Wikis, Collaborative Annotation Systems, Semantic Web, and Linked Open Data.

Annotation Models

1. Two lightweight annotation models are developed. One is tailored for the document-based scenario, representing document-based annotations. It is concerned with tying annotations to parts of text documents. The other model is tailored for the resource-based scenario. It focuses on how to represent user confidence and the degree of truth in annotations. Both models can be combined. This contribution relates to hypothesis H1.

Recommenders

2. Several recommenders are developed. Goal-directed annotation recommenders are key to improving annotation-based functionality and improving user experience, as unassisted creation of annotations is tedious. This contribution relates to hypothesis H2.

Personalized Views

3. An approach for building personalized views based on the potentially conflicting opinions of users is developed. Building on user similarity, the system can weight other users' opinions concerning specific facets of resources available in the system, and present a personalized weighted view of the data available in the system. This contribution relates to hypothesis H3.

Wrapping Web Data Sources

4. An approach for online wrapping Web data sources as Linked Data is presented. A prototype implementation of this approach supplies rich ontologies and instances for both the document-based and the resource-based system developed in this thesis. This contribution relates to hypothesis H4.

5. In an evaluation combining all three systems, a large amount of testing data was collected, and the validity of the personalized views approach has been shown.

1.4. Thesis Overview

This thesis consists of five parts. The first part presents the **Motivation** behind my work. The second part describes **Foundations** including related work. The third and longest part, the **Approach**, explains the approaches pursued for the two scenarios, the prototype software implementations in these scenarios, and software used in both approaches. The fourth part is an **Evaluation**. The thesis ends with a **Conclusion**.

Part I: Introduction

In Chapter 1, a motivation for handling **ontology-based annotations** is given. Challenges in pursuing this approach are listed. The **two scenarios** the work is applied in get introduced. Section 1.2 presents the **aim of the research**, and lists **research hypotheses**. In Section 1.3, the **contributions** of this thesis to several fields are outlined. Section 1.5 lists related own publications, supervised master theses, and gives acknowledgements to co-workers.

Part II: Foundations

In Chapter 2, **concepts, approaches, and technologies** this thesis is based on are explained. For the approaches this thesis expands, **related work** is given, and differences highlighted.

Part III: Approach

Part III is the main part of this thesis, detailing **three approaches** to different challenges. Two of these approaches are targeting the two scenarios presented in the motivation chapter, each accompanied by a software prototype tailored for that scenario. In Chapter 3, the **document-based scenario** is addressed; a Semantic Wiki prototype, *Kaukolu*, is presented. Chapter 4 is concerned with the **resource-based scenario**. Here, the *Skipforward* annotation system is presented. Chapter 5 presents an approach to **wrap non-semantic data sources** and expose the data using Semantic Web techniques. The concrete data source presented, *DBTropes*, can be used by the other two software systems, as is done in the evaluation that follows.

Part IV: Evaluation

While some aspects specific for the individual approaches are addressed by evaluations mentioned in the respective chapters, Part IV presents an evaluation that combines all three systems into one evaluation scenario. The approaches evaluated were born out of analyses of (shortcomings of) existing technology, application of collaborative approaches, and iterative improvements of existing technology. User questionnaires were employed to assess the importance of these improvements from an end user perspective instead of a researcher perspective. The major part of the evaluation is concerned with hypothesis H3, evaluating the approach to subjective annotations presented in this thesis.

Part V: Conclusion

The thesis ends with a **conclusion** and **outlook**. The outlook focuses on possible future applications of the approaches and highlights areas in which the approaches could be extended.

1.5. Acknowledgements

The work presented in Chapter 3 (the document-based scenario) was part of the Mymory and NEPOMUK projects at DFKI. I would like to thank my colleagues in these projects, namely Ludger van Elst, Georg Buscher, Sven Schwarz, Heiko Maus, Leo Sauermann, Gunnar Grimnes, and Benjamin Adrian, as well as the students who worked on the prototype software (Matthias Käppler, Paul Pichota, Artun Subasi, Lizhen Qu, Benjamin Mock, Florian Mittag, Li Gui, Moritz Plößl, Thomas Friedel, Dominik Heim, Ralf Biedert) for their help. Several papers have been published in the course of the research [KSEB08, vEKS+08, Kie06]. The chapter is partially based on these results.

Chapter 4 (the resource-based scenario) is based on work that many people contributed to. I would like to thank my colleagues Sven Schwarz, Heiko Maus, and Rafael Schirru as well as the students Florian Mittag, Michael Lörscher, Stephanie Schuster, Markus Fuchs, and Injy Hamed for their help. Several papers have been published in the course of the research [KM11, KH13], and two master theses have been written [Mit08, Ami12]. The chapter is partially based on these results. The topic was part of the iGreen project.

In context of the work presented in Chapter 5 (wrapping external ontology sources), I would like to thank Gunnar Grimnes, who helped with the DB-Tropes visual design and papers, and Artun Subasi, who implemented and documented the interlinking component presented in this chapter. The work is topic of several publications [KG10, Den12] and was part of the THESEUS research program.

Text Styles

Throughout the thesis, these text styles are used:

- RDF properties such as *owl:sameAs* are presented in *italic text*.
- Resources (plain RDF resources as well as RDFS classes) such as rdfs:Class use sans serif text.
- Code (pointers) or constants such as `HashMap` are shown in `typewriter font`.

II

Foundations

CHAPTER 2
Background and Related Work

In the following, the concepts and technology that represent the background of this thesis are introduced. First, the concepts of *metadata* and *annotation* are discussed, then the ideas behind the Semantic Web, Semantic Desktop, and Semantic Wikis are introduced. The chapter ends with a discussion of Collaborative Annotation Systems, and Annotation Recommender functionality.

2.1. Metadata and Annotations

Metadata

For search and lookup purposes, systems storing information (be it a library and a traditional paper-based index or an IT based system) typically rely on *metadata*—data about data. According to the Digital Futures Group (DFG) at the Library of Congress[1], metadata can be divided into three categories:

- **Descriptive Metadata** describing the content of the resource in question, e.g., a list of topics a book is concerned with, a textual summary, its title,
- **Administrative Metadata** describing provenance and maintenance information of the resource, e.g., archival date or access rights of a book in a library,
- **Structural Metadata** linking individual parts of the resource together, e.g., providing information on what pictures belong to what document.

The boundaries between these categories are fluent; some examples given by the DFG belong to all three categories[2].

In more recent (digital) use cases, one additional metadata category became more and more important: **Use Metadata**, which represents data collected from or by users. This includes information about the (past) use of the resource such as number of accesses, or user-generated comments.

[1] `http://www.loc.gov/standards/metadata.html` – accessed December 2013
[2] `http://www.loc.gov/standards/metable.html`

Capturing of Information Content gets classified in [KS00]:

- **Content Independent Metadata** that does not depend on the content of the resource it is associated with. This includes the location of storage of the data, and its last modification date.
- **Content Dependent Metadata** that depends on the content of the resource it is associated with. This includes the length of a text or the dimensions of an image. This can get categorized further into **Direct Content-based Metadata** (such as an automatically derived full text index) and **Content-descriptive Metadata** (which describes the resource's content but is not derived directly from it).

Other terms often used for metadata classification are **Intrinsic Metadata**, which is used for direct content-based metadata, and **Extrinsic Metadata**, which means content-descriptive metadata typically and has a focus on manual generation.

Other partially orthogonal dimensions of metadata exist as well.
[SSSS01, Han05] classify metadata according to the following dimensions:

- **Formality**—ranging from textual comments to annotations using rigid categorization schemes.
- **Containment**—whether the annotation is embedded into the resource it annotates, e.g., as additional markup in an HTML document, or is stored externally, merely pointing at the annotated passage (e.g., using XPointer for HTML/XML documents).

Annotations

Annotations are a specific variant of metadata. Annotations represent metadata typically being specific to a resource context: for example, an underlined passage in a book represents an annotation, as well as information on who modified what line of a text file last in a source control management system used for collaboratively editing file-based projects such as software projects. Web pages can get annotated as well (see [KSP03] for an early project in this direction); the ubiquitous user comments in Web 2.0 Websites can be seen as annotations for the resource they are associated with. To give an example, Soundcloud.com allows users to annotate/comment specific parts of the music pieces available on the platform.

2.1.1. Tagging

With the advent of Web 2.0, *Tagging* has become a widespread way of annotating resources. These Tags can be used in search or for exploring datasets. Mostly,

Tags are implemented by ways of letting the user who created the resource assign arbitrary strings to it. With Flickr.com as one of the early implementors, *Tag Clouds* were omnipresent on Web pages, providing a somewhat standardized way of giving an overview of currently relevant keywords for explorative purposes. However, for general navigation Tag Clouds are less suited, which lead to decline of their usage.[3]

Tag Clouds

The term *Folksonomy*, a portmanteau of *folks* and *taxonomy*, coined by Thomas Vander Wal [Wal07], represents the set of Tags used by a community. Folksonomies exhibit interesting properties: In contrast to taxonomies, no predefined relationships between Tags exist—this also means that there is no explicit hierarchy. Tags typically do not carry any additional information—there is no comment describing a Tag, or any other literal value associated with a Tag instance. It has been shown that with a sufficiently large user base, a stable core vocabulary can form [RHS09]. There have been efforts to build logically sound taxonomies or even ontologies from folksonomies [TKH11].

Folksonomy

Tagging systems are used for descriptive metadata almost exclusively: For structural and administrative metadata, objects and/or literals need to be stored in the metadata typically. Annotation formality is low as semantics are often somewhat ambiguous. Containment can be handled in multiple ways; whereas Flickr and other services allow users to assign Tags to resources, other services like Twitter.com promote embedding Tags in the actual content.

Classification of Tagging

Tagging implements an *open vocabulary* approach since any user may introduce new Tags whenever deemed convenient. A comparison of the controlled vocabulary and open vocabulary approaches is presented in [Mat04]. Ontology-based approaches such as described in the following mostly implement controlled vocabulary approaches.

Open Vocabulary

2.1.2. Ontology-Based Annotations

Whereas in tagging systems Tag instances are implemented and represented using strings usually, ontology-based systems use instances of ontology classes (from here on: *annotation classes*) in general.

[3] Flickr later went as far as stating "sorry about the Tag Clouds" in their 2006 acceptance note for the "Best Practices" Webby Award.
`http://www.webbyawards.com/press/archived-speeches.php`

Characteristics

This has a few implications:

- One annotation (instance) is always associated with an annotation class.
- Annotations can use hierarchy as annotation classes can be hierarchically structured.
- Annotation classes may carry further information, setting them in context with other classes (see SKOS, the *Simple Knowledge Organization System* [MB09], for a number of relevant relationships).
- Annotation instances may carry further information such as a plain text comment or references to other instances or resources.
- In collaborative systems, unless users are allowed to modify the ontology level, they are not able to create new annotation types.
- By analysis of Tag usage in tagging systems, semantics can be extracted post-hoc. In ontology-based systems, semantics are predefined.

Comparison with Tagging

In comparison to the tagging approach, ontology-based annotations have pros and cons:

- Metadata per annotation class means more possibilities for machine processing from the start.
- Semantics of annotation classes can be made unambiguous through comments and explicit relationships to other classes; in tagging systems, semantics of Tags can be subjective or unclear.
- More metadata per annotation class also means that extending the annotation ontology is more work intensive than extending the vocabulary in a tagging system.
- Since annotation classes tend to be more heavyweight than Tags, ontology-based systems are prone to be less flexible.

Classification of Ontology-Based Annotations

Ontology-based systems can be used for descriptive, structural, administrative, and use metadata alike (cf. Section 2.1). However, the implementation will vary a lot for each use case. Sources and consumers of these annotations are quite different: for example, structural metadata will likely be generated and consumed by software only. In this thesis, the focus lies on ontology-based descriptive annotations as well as annotations containing use metadata. However,

some structural and administrative metadata will be handled using ontology-based annotations as well. Annotation formality is high, as annotation ontologies usually get created by domain experts. Concerning containment of annotations, ontology-based annotations mostly get stored separately from the resource they annotate. This is due to the fact that ontology-based annotations need elaborate data structures which are unwieldy to embed in, for example, plain text documents. However, technically embedding is possible as well; this can be done by means of RDFa (*Resource Description Framework in Attributes*[4]) and other standards.

In the context of this thesis, annotations are *instances* of ontology concepts assigned to (parts of) documents or general information resources (such as books or people).

2.2. Semantic Web

Machine-Interpretable Information

The World Wide Web and technologies such as Web search engines make very large amounts of information available at your fingertips. However, combining information elements, evaluating information credibility, condensing information, etc., stays impossible for machines: The HTML technology underlying the WWW allows layouting and displaying information on screen, but the information itself stays mostly unreadable and incomprehensible for machines. Therefore, many tasks that could potentially be carried out by machines still need to be done by humans currently:

- Finding the correct hits in a Web search (not merely presenting a list of potential matches).
- Combining facts from different sources (in a semantic form, not merely by concatenating text snippets).
- Managing trust and weighting statements from different sources and users.
- Combining services such as navigation and ticket buying services.

For these tasks to be carried out by computers, technology for making the contents of the Web semantically available on a machine level is being developed.

[4] `http://www.w3.org/TR/xhtml-rdfa-primer/`

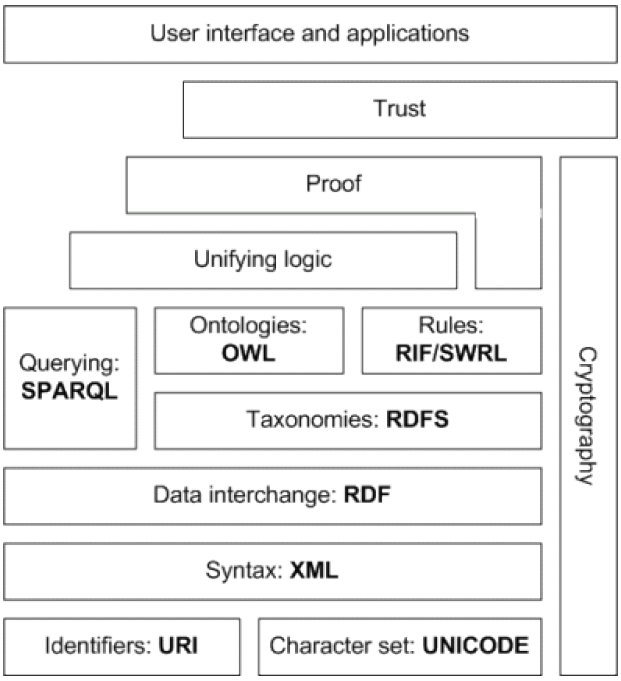


Figure 2.1.: The Semantic Web stack. [Obi]

Semantic Web Vision

The Semantic Web vision aims at putting this into practice. A number of technologies was developed for making information available to computers on a semantic basis. In Figure 2.1, the Semantic Web Stack is shown, starting with base technologies for data representation such as URIs and unicode, via non-semantic document formats (XML), to semantic data formats (RDF/S, OWL) and rule/querying languages working on these formats (RIF/SWRL/SPARQL). The layers on top of these ("unifying logic", "proof", "trust") have not been standardized yet, and, if needed, are typically implemented in an application specific way in applications available currently.

Basically, the idea behind the Semantic Web is to make the data that Web services need available in clearly defined and machine readable formats. In this case, RDF is used as data representation format; the data made available as RDF is structured according to schemas/ontologies formulated in RDFS or OWL. Due to the explicit subclass/subproperty properties available in these languages, some semantics can be derived immediately as long as appropriate cross-references to vocabularies known to the application are given.

A simple ontology language for the Semantic Web is the set of RDF and RDFS, together denoted as RDF(S). In the following, the RDF(S) language, a Semantic Web standard, will be described.

RDF(S) - Resource Description Framework (Schema)

Standard Data Exchange Format

The *Resource Description Framework* [rdf] is a standard for data exchange in the (Semantic) Web. In the following, the RDF(S) data model will be described, and it will be compared to the model known from object oriented programming (OOP). RDFS *classes* get organized in a class hierarchy just as is the case in OOP. These classes have *properties* (in OOP: member variables), allowing to populate *instances* of the classes with data. RDFS supports XML Schema datatypes for defining what (literal) data is allowed for what properties. In contrast to standard OOP, properties are organized in a *property hierarchy*, subproperties being more specialized than their superproperties. Also, there may be multiple values associated to one property in an instance of a class. In OOP this could be modeled by member variables being an array containing a number of values of the designated type. Another specialty of RDF(S) is that multiple inheritance is allowed: An instance may be instance of more than one class; a class may have multiple superclasses.

Schema Language

RDFS is the Schema language associated with RDF. In RDFS, classes including their hierarchies, and properties and their ranges (either class instances or literals) can be defined.

Graph

All data of an *RDF model* is ultimately represented as a graph or a set of *RDF triples*. A triple or *statement* is essentially an edge in the graph. It consists of subject, predicate, and object, subject being the originating node (typically represented by a URI), predicate denoting the type of relation, and object being the target node (or literal). See Figure 2.2 for an example that shows the same data first using an RDFS class/instance-based view and then using RDF triples. Note that here we only want to illustrate the mapping from RDF(S) to RDF triples. Reusing vocabulary or explaining namespaces is not covered here.

Existing Ontologies

A rich set of (RDFS) ontologies is available ([Dub11, BM07], etc.). Each ontology defines the *vocabulary* of a certain domain. The main idea is that everybody is free to take a readily available vocabulary and extend/specialize it for the current needs—applications can work on this data even if they only know the basic vocabulary. Some approaches define top-level vocabularies, so-called *foundational ontologies* such as SUMO [NP01], OpenCyc[5], or DOLCE [Gan07]. These ontologies define the most basic concepts from the real world such as classes of occurrences and things, physical objects and processes, and space and time. This is where ontologies blend with techniques used in artificial intelligence.

[5] `http://sw.opencyc.org/`

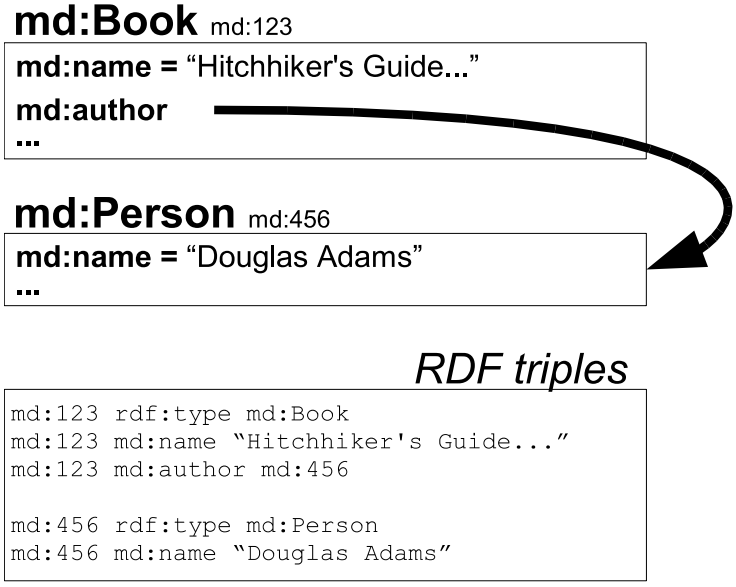


Figure 2.2.: RDF(S) instances view vs. RDF(S) triple statements.

Triple Stores

On a computer, RDF data is typically managed and accessed by a *triple store*. Concrete implementations are, for example, Apache Jena[6] and Sesame[7]. While the history of triple stores is somewhat different than that of (object-)relational databases, triple stores represent a form of database; triple stores can be seen as NoSQL graph-based databases. Mappings between the RDF data model and the table-based models of relational databases can be built [SK06].

Reasoning and Query languages

Inferencing Knowledge

Reasoning—in the domain of ontologies—means the act of inferencing knowledge from something already known. The classical example is "All men are mortal. Socrates is a man. Therefore Socrates is mortal.".

In pure RDFS, only limited subclass/subproperty reasoning is available. One peculiarity of RDFS has to be mentioned. In contrast to, for example, XML Schema, RDF Schema cannot be used to "validate" RDF. Any RDF data is "valid"; RDF Schema is merely there to infer additional statements. For example, if in RDFS there is a class Human with a property *has_car* and a class Dog

[6] http://jena.apache.org/
[7] http://www.openrdf.org/

with a property *has_food_bowl* and, as RDF data, the statements human5 *type* Human and human5 *has_food_bowl* bowl3 exist, this would not classify as invalid; instead, an RDFS reasoner would infer that human5 is of type Dog, *too*. Statements of the type "Instances may not be of type both Human and Dog" are not possible in RDFS; this is available only in OWL[8].

Rule Languages

Rule languages such as SWRL [Con04] add more expressivity to RDF: for example, with rules one can formulate terms such as "every instance of (a subclass of) the class Male that has a property *has_child* is also an instance of the class Father. In the context of RDF, rules are often used to define *inverse properties*: "if an instance A has a property *has_child* pointing to instance B, add instance A to the values of instance B's *child_of* property"—RDF itself does not provide ways to define inverse properties.

Querying

Querying means retrieving data from a dataset by specifying clauses that data has to satisfy. For RDF, typically SPARQL [PHS13] is used for querying. The SPARQL standard standardizes both an RDF query language and the query protocol.

The SPARQL query language looks similar to SQL[9] at first glance. It allows to extract information from RDF graphs, extract RDF subgraphs, and can construct new RDF graphs based on the query.

Linked Open Data

Publishing Data

Linked (Open) Data (or *Web of Data*) [BL+09] is both a method of publishing data in a machine-interpretable way and the set of information repositories using this method. Its technical foundations are very similar to those of the World Wide Web; however, instead of serving machine-renderable HTML, RDF(S) is served to clients that discern themselves from Web browsers by using HTTP content negotiation and HTTP redirection. Thus, clients are not Web browsers typically, but Linked Data-enabled software that can have arbitrary frontends: for example, mashups exist that have been implemented as smartphone apps, using data made available by third parties[10], as well as mashups that expose their services using a normal Web frontend (e.g., [BCG07]).

Four Principles

The four principles Linked Data build on have been outlined by Tim Berners-Lee in his *Design Issues: Linked Data* note[11].

[8] `http://www.w3.org/OWL/`

[9] The *Structured Query Language* typically used for querying relational databases.

[10] `http://semtech2011.semanticweb.com/sessionPop.cfm?confid=62&proposalid=3974` – "Building Mashups for the Linked Data Cloud" tutorial

[11] `http://www.w3.org/DesignIssues/LinkedData.html`

1. Use URIs to denote things.
2. Use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
3. Provide useful information about the thing when its URI is dereferenced, leveraging standards such as RDF, SPARQL.
4. Include links to other related things (using their URIs) when publishing data on the Web.

Chicken and Egg Problem

Linked Open Data was born out of the realization that the Semantic Web effort got stalled by a chicken-and-egg problem: Technology and standardization did not evolve as fast as possible since there was little data publically available; in turn, data providers had little incentive to make their data available using the little known standards and technologies developed in the Semantic Web community.

Linked Open Data focuses on bringing Semantic Web technologies to the public and employing them on Web scale. The Semantic Desktop paradigm, explained in the following, aims at using Semantic Web technology on a single user's workplace and in small user groups.

2.3. Semantic Desktop and Personal Document Management

Scenario

The Semantic Desktop scenario is focused on a knowledge worker using his desktop computer. On the personal computer, having the right information at hand at the right time is of high importance. One characteristic of this scenario is the following: rather than searching for information previously unseen by the user—as is the case in (Semantic) Web scenarios—the knowledge worker will typically search for information he has seen previously, or even information that was created by him at some point in the past. Thus, the task of finding in the scenario of the Semantic Desktop is mostly a task of finding *again*.

Resources

On the conventional desktop, lists of resources are omnipresent, e.g., lists of files, (alphabetically sorted) lists of people, or (chronologically sorted) lists of visited Websites. However, information such as *This file is of interest in context with project X*, or *This Website was read while studying topic Y*, is neither captured nor available by other means. While users can work around these limitations by, for example, using naming conventions for files, and using elaborate Website bookmarking hierarchies, it is obvious that automated support for interconnecting information resources is desirable.

Semantic Desktop Vision

The Semantic Desktop aims at providing more structure and interlinks between information resources. The alphabetically sorted list of people mentioned above becomes a list of people relevant in the context of the document the user is currently looking at—relevant because the people in the list have some semantic connection to the project that is associated with the document. So, in the Semantic Desktop it must be possible to explicitly represent a number of types of resources such as Persons, Projects, Documents, etc., as well as associating these resources with one another.

Related Work

One of the first prototypes implementing Semantic Desktop ideas was *Haystack* [QHK03]. Haystack was an integrated software that aimed at combining standard desktop applications in one software, and providing Semantic Desktop features.

At DFKI, several projects have explored and implemented key Semantic Desktop concepts.

The EPOS project (*Evolving Personal to Organizational Spaces*, 2003-2005) investigated how to connect an individual knowledge worker's workspace to a shared organizational space [SDE+06]. The premise is that people prefer to work in an environment customized to their needs and liking. Categorizations and other structuring facilities on an organizational level will not fit individual user's needs. EPOS developed ways of connecting both worlds, having them benefit from each other. The implementation included an early version of a Semantic Desktop environment called *Gnowsis*.

The Mymory project (2006-2008) explored capturing user context and user attention [vEKS+08, KSvEB08, KSEB08]. For modeling user context, Semantic Desktop components were used. A Semantic Wiki was used as a core part of the project, serving as a document repository. The Wiki had facilities to capture user attention information using an eyetracker.

The NEPOMUK project (2006-2008) developed a full-fledged Semantic Desktop with social functionality [BGGS11, GHM+07]. In the course of the project, several Semantic Wiki approaches were evaluated. These Wikis connected with the core data structures of the Semantic Desktop system, letting the user annotate Wiki texts with concepts from his personal information model and vice versa.

2.3.1. Personal Information Model (PIMO)

The need for a way to model (personal) knowledge structures in the Semantic Web setting was recognized quite early. [Guh96] described the Meta-Content Framework, later leading to RDF (see Section 2.2), and realized "*rich, standard,*

Concepts in Knowledge Work

structured, extensible, compositable descriptions of information organization structures as the core of information management systems". In practice, knowledge workers use a lot of information resources every day; these resources can be grouped into a few types. To list some examples:

- People and organizations as well as associated contact information
- Projects
- Topics such as knowledge management or semantic desktop
- Events
- Tasks
- Documents or files on a computer

PIMO Overview

In Semantic Desktop-related projects at DFKI, the PIMO (*Personal Information MOdel*, originally developed in the EPOS project [SDE+06]) is used to model and relate resources of these types. The PIMO aims to provide a sound formal basis for modeling these concept (instances) but also provides ways to add informal knowledge to the knowledge base of the user. The idea is to have a unified model of all the resources the user interacts with, and being able to create relations between these resources and concepts. This can be done separated from the applications that are typically used to manipulate these resources (e.g., a desktop calendar, address book, or text processor). Some functionality concerning the PIMO such as associating a text file with the concepts mentioned in it can be done automatically using information extraction techniques (see Section 2.9).

2.3.2. User Context

Goals

User context was a research focus in the Mymory project [vEKS+08]. The user context is concerned with questions such as *What project is the user currently working on?* and *What topics are currently relevant to the user?*. A preliminary for being able to model user context using the following approach is to have a user PIMO (see Section 2.3.1). In the project, user interactions with his desktop software is captured. To achieve this, a number of plug-ins for typical desktop software such as Web browsers, mail clients, and other programs such as file system change listeners and mouse and keyboard observers were written. The output of these programs, e.g., information about the user opening a file with an editor, or switching program windows, is called NOPs (*Native OPerations*).

Native Operations

These NOPs get routed into the *context elicitation framework* which matches the NOP data against a set of rules that are partially autogenerated from the user's PIMO. Aggregation of NOPs results in an (updated) PIMO concept activation vector. This activation vector represents the user context. It changes over time, and further components deal with detecting *context switches*. Every distinct context is assigned a URI which can be used as a simple identifier of a context.

Attaching Context

In the Mymory project, user context can be attached to resources and annotations, and can be used in search, for ranking information, giving context dependent recommendations, and generating new associations.

A detailed description of the user observation and context elicitation framework can be found in [Sch10, SRB03]. Parts of the frameworks are available as open source[12].

2.3.3. User Attention

While User Context components can detect what documents the user is currently studying, it is difficult to detect what parts of that document are of particular interest to the user. However, this user attention information is crucial for (re-)finding information. In the Mymory project, eyetracking was investigated as a means of detecting user attention within larger documents. Using current eyetracking hardware based on infrared cameras in the monitor, the eye fixation position can be detected with an accuracy of some millimeters. With some postprocessing of the raw fixation points, further information about the user's reading process can be derived: for example, detecting whether the user just skimmed some text, or whether the user had problems reading some text part, becomes possible.

Eyetracking

Re-Find and Ranking

User attention information can be attached to resources, similar to User Context information. In turn, user attention information can be used in search (to re-find information previously read, or focus search results on read passages), for ranking information (rank information that has been read by the user before higher), and generating new associations (by analyzing the text of read passages).

Conceptually, user attention can be stored in a Semantic Wiki. This way, user attention information can be handled in a similar way to other information, and even get enriched with, for example, User Context information.

More information about measuring user attention and using user attention to improve search and retrieval can be found in [Bus10].

[12] `http://usercontext.opendfki.de/`

2.4. Wikis and Semantic Wikis

In the following, an overview over the (Semantic) Wiki idea and about typical features will be given.

2.4.1. Wikis

Wiki Standard Features

Wikis are lightweight Web-based tools that allow a group of people to collaboratively author information. Standard features of Wikis include:

- Storage of a set of *Wiki pages* (typically text in some Wiki markup syntax); each Wiki page has a name
- Rendering pages as normal HTML, linking to other pages—sometimes automatically linking by page name
- Editing of pages
- Versioning of pages and associated functionality (history view, diffs between versions, RSS feeds[13], etc.)

These main features a Wiki provides are simple but allow flexible use of the Wiki for a number of different purposes: for example, the very basic Wiki idea of editing a text page allows both editing a document and discussing with other Wiki users—by means of adding comments below other people's comments.

Wiki Use Cases

Current applications of Wikis range from open encyclopedias such as Wikipedia to collaborative information spaces for both open communities such as open source software projects (e.g., `http://wiki.mozilla.org/`—even software project management software such as Trac[14] feature Wikis for documentation and information exchange) and closed communities such as company intranets.

A large number of Wiki implementations exist, of which many extend on the features presented above. Some implementations focus on special requirements such as scaling to a high number of users and/or pages, or pursue other ideas such as providing a single-user knowledge space as a desktop application.

[13] *Rich Site Summary*, a standard for publishing information about data updates, see `http://en.wikipedia.org/wiki/RSS`

[14] `http://www.edgewall.com/trac/`

2.4.2. Semantic Wikis

Definition

According to Wikipedia, "A semantic wiki is a wiki that has an underlying model of the knowledge described in its pages. Regular, or syntactic, wikis have structured text and untyped hyperlinks. Semantic wikis, on the other hand, provide the ability to capture or identify information about the data within pages, and the relationships between pages, in ways that can be queried or exported like a database."[15]

Features

To implement this, features of Semantic Wikis include:

- Typing of pages and links between pages using well-defined types
- Text annotations
- Import/export of data formalized in RDFS/OWL/CSV
- User support when editing—providing templates, etc.
- Ontology support—verifying data stored in the Wiki, etc.
- Enhanced search that can make use of page/link types and annotations

Machine-Readability

The overarching aim of Semantic Wikis is to make the underlying information structure available in a machine-readable way. In turn, this allows using the information stored in the Wiki in external applications as well as for improving search, navigation, and general user and contributor experience within the Wiki itself. Other possible benefits include formal verification of facts entered in the Wiki, and editing help such as providing templates for new resources.

Example

In the following, an example using the well-known Semantic Wiki *Semantic MediaWiki* (SMW)[16] [KVV05] will be given. SMW is an extension of *MediaWiki*[17], the software used by Wikipedia. Metadata associated to a Wiki page may point to other resources, but in SMW, literals are allowed, too. Also, metadata is entered directly into the Wiki text, and does not necessarily have to adhere to a schema. SMW supports multiple datatypes such as coordinates and temperatures, along with conversion between different unit scales.

[15] `http://en.wikipedia.org/w/index.php?title=Semantic_wiki&oldid=561796523` – accessed June 2013
[16] `http://semantic-mediawiki.org/`
[17] `http://mediawiki.org/`

The following is SMW example markup for a Wiki page about *Amsterdam*[18].

> '''Amsterdam''' is a city in the [[Located in::Netherlands]]. In [[Year::2010]], the population was [[Population::783,364]] and the area [[Area::219 km^2]]. It rains on [[average rainy days::234]] days per year on average.
> [[Category:City]]

Notable differences to standard Wiki markup is that here, (i) links to other pages (resources) get *typed*, and (ii) literals such as the population count get typed as well – in standard Wiki markup, the population count would not be marked up, resulting in the number "783,364" not carrying any semantics for the Wiki implementation.

Figure 2.3.: A Wiki page about *Amsterdam* as rendered by Semantic MediaWiki.

SMW creates a page as seen in Figure 2.3 from this markup. Also visible in that screenshot is standard Wiki functionality (e.g., an edit link, and a link to the page history). The upper part of the page content is rendered directly from the markup. Then, the category link follows. Finally, the "Facts about" box represents the main functionality that sets apart SMW from standard MediaWiki: There, the facts given in the markup are shown again, in a well-structured form, and allowing to download the data in RDF. Additional navigational links are present; for example, the link right from "Located in: Netherlands" leads to a

[18] `https://semantic-mediawiki.org/wiki/Demo:Amsterdam` – accessed December 2013

list of all pages that exhibit that fact (i.e., all pages that are about locations in the Netherlands).

Related Work

Related work for Wikis gets listed in the following, along with comparisons to the *Kaukolu* Semantic Wiki approach presented in Chapter 3.

In most traditional Wikis, the idea of metadata typically only appears in a very technical way: for example, in *JSPWiki*[19], metadata is added directly into the Wiki text using special Tags, and mostly serves the purpose of implementing access control. In *SnipSnap*[20], labels may get attached to Wiki pages, serving mainly as a categorization scheme.

One of the very first Semantic Wikis, *Platypus*[21], added RDF(S) and OWL metadata to Wiki pages. In contrast to more modern approaches, metadata had to be entered separately from Wiki text and relates a Wiki page to another resource; thus, metadata can be transformed into a list of *related pages* that can be shown along with the actual Wiki page.

Another early Semantic Wiki, *Rhizome*[22] [Sou04], built on a framework that adapts techniques such as XSLT and XUpdate to RDF. In essence, RDF is used throughout the framework, and RxSLT (an XSLT variant adapted for RDF) is used for transforming queries' results to HTML or other output formats. Similar to the Platypus approach, page metadata has to be entered separately from the page. While the approach is very interesting from a technical point of view, the implementation requires a lot practice with the underlying techniques.

IkeWiki[23] [SGW05] is Semantic Wiki supporting OWL ontologies. It supports inferencing when typing links and relies on JavaScript–based features for supporting the user which helps quite a lot when adding semantic information. The KiWi (Knowledge In A Wiki) project [KSB+10] later built on IkeWiki ideas, evolving into a "Platform for building Semantic Social Media Applications", additionally focusing on recommendations (cf. Section 2.6).

In [BSVW12], an overview of Semantic Wikis and their concepts is given.

SweetWiki [BGE+08] uses RDFa[24] as technical implementation of its annotations and therefore potentially supports associating semantic statements to text in a similar way as does Kaukolu. The possibilities of this text/semantics connection seems not to be used in search beyond what is known from normal Semantic Wikis though (i.e., search results on page level granularity, or generating

[19] http://jspwiki.apache.org/
[20] http://snipsnap.org/
[21] http://platypuswiki.sourceforge.net/
[22] http://www.liminalzone.org/Rhizome
[23] http://sourceforge.net/projects/ikewiki/
[24] http://www.w3.org/TR/xhtml-rdfa-primer/

tables based on RDF result data as search results).

OntoWiki [TFH10] is a modern Linked Data-enabled Semantic Wiki. It supports RDFa annotations and editable forms as well as custom visualizations such as maps.

Wikidata[25] is "a free knowledge base that can be read and edited by humans and machines alike. [...] it centralizes access to and management of structured data [...]". Wikidata was started in 2012 and aims at providing a shared data pool for the individual language-specific Wikipedias: Previously, Wikipedia infobox information was duplicated for each language-specific Wikipedia; Wikidata is an effort to de-duplicate that information, simplifying maintenance and improving knowledge re-use between the individual Wikipedias. The structure of the data available via Wikidata is similar to the data that would be available if Wikipedia used Semantic MediaWiki directly; DBpedia [LIJ+14] differs insofar as the data available through is has been enriched with additional semantic information and has a stronger focus on ontology re-use.[26]

Comparison

None of these systems support detached annotations, attaching user context information to the semantic content, or fine-grained (annotation/paragraph-level) semantic search, as Kaukolu does.

2.5. Collaborative Annotation Systems

Collaborative Annotation Systems or *Social Annotation Systems* are systems that allow a group of people to annotate resources. The aim of this is to make the content of the system better accessible and searchable.

As presented in Section 2.1, annotations can get realized in various ways, for different purposes. Collaborative annotation systems typically are Tag-based and use an open vocabulary, are ontology-based with mostly closed vocabulary, or use a mix of both approaches.

Tag Recommenders

One of the practical challenges in annotation systems is helping the user with creating annotations. See Section 2.7 for an overview of Tag recommendation approaches used in a number of different annotation systems.

Related Work

Examples for collaborative annotation systems and differences to the prototype systems *Kaukolu* and *Skipforward* presented in Chapters 3 and 4 are:

Revyu.com [HM07] allows users to submit reviews which can be tagged with keywords. Absolute ratings can be given to items. Metadata is available as RD-

[25] `http://www.wikidata.org/`

[26] As of 2013; it is expected that with all projects being in development, these facts might change quickly.

F/Linked Data; however, the tagging-based approach gives shallow metadata only. In contrast to Skipforward, (formalized) discussions about annotations are not supported, and there is no personalization.

DBin [TM07] is similar to Skipforward but more generic and heavyweight. It comes with its own messaging API, a plug-in architecture for its user interface, and needs dedicated metadata servers and a Java client whereas in Skipforward no server component is needed—the client, due to its Web-based nature, can be run locally or as a shared installation on a server in case users do not want to install special software. Metadata sharing in DBin is based on a complex subgraph mechanism whereas Skipforward uses a simple namespace-based approach similar to Linked Data.

Bouillon[27] implements a "peer-to-peer Wiki" using Jabber/XMPP. Wiki text parts given a good rating are weighted high for the user's friends (i.e., people in the user's contact list) and vice versa. Thus, there is no 'canonical' view on the contents of this Wiki but there exist many subjective views, one for each 'community', which is similar to the idea Skipforward pursues. *Personalized relevance* [HM06] is a similar concept.

In terms of recommendation functionality, Skipforward implements a semantic hybrid filtering model. Similar approaches are discussed in [Paz99] (the recommendation channel concept of Skipforward is similar to the *Collaboration via content* approach outlined in that paper) and [CC06] (clustering users based on domain concepts they are interested in—possible but not implemented in Skipforward).

sobooks.de is an upcoming books reviewing system with social media integration such as Twitter. It allows starting discussion threads linked to individual paragraphs in a book. So far, it does not feature any semantic features though; sobooks is comment-/like-based.

ALOE [MS07] is "A Socially Aware Learning Resource and Metadata Hub". It allows users to share and tag resources such as Website links, text documents, or arbitrary files. Similar to Skipforward, it features item and annotation recommenders.

BibSonomy [HJSS06] is a "social bookmark and publication sharing system". It heavily focuses on providing a database of research papers and, as such, has features such as BiBTeX integration.

Comparison

None of the approaches outlined above feature personalized views of content-based annotations as does Skipforward; none provides explicit (fine-grained/feature type-level) user similarity.

[27] `http://bouillon.math.usu.ru/index.html%3Fp=45.html` – unfortunately, as of 2013, the demonstration server is offline.

2.6. Recommender Systems

One very important functionality in systems that store information about items is the functionality to *recommend items to users* in a concise and simple way, not requiring the user to sift through dozens or hundreds of textual reviews or demand from him to compare item characteristics manually. This *item recommendation* is a major component of services such as Amazon[28], YouTube[29], Last.fm[30], or Pandora[31]. Typically, item recommendation has the goal to recommend items that the user will *like* (e.g., a book or song he will like) but he does not know yet. In more mathematical terms, recommendation is about estimating the user's ratings for items he has not rated yet. The items that have the highest estimated rating get recommended to the user.

Recommender Approaches

Different approaches are possible to implement the rating estimation component. Basically, two major approaches (plus their combination) exist:

- **Collaborative Filtering** (CF) tries to predict user ratings by looking at the ratings of other users ("Users who rated items similar to you also liked these items...").
- **Content-Based Filtering** (CBF) generates recommendations by looking at characteristics of items the user has rated high, and recommending items that share these characteristics ("You seem to like rock music with male singers. Have a look at these songs...").
- **Hybrid Approaches** combine these two approaches or employ other data sources in the recommendation process.

In the following, these approaches get explained in more detail.

2.6.1. Collaborative Recommendations

In order to estimate the rating of the user concerning a yet not rated item, collaborative filtering analyses existing ratings of the user and similar users (users that gave similar ratings to items in the past). The assumption here is that users who gave similar ratings to items in the past will continue to do so in the future. Of the existing recommender systems mentioned above, most implement collaborative filtering (Pandora is the exception).

[28] http://www.amazon.com/
[29] http://www.youtube.com/
[30] http://last.fm/
[31] http://www.pandora.com/

Pearson Correlation

For calculating user similarity, typically the Pearson product-moment correlation coefficient[32] is used. This correlation coefficient is defined as the covariance of two variables (e.g., product ratings by two different users) divided by the product of their standard deviations. The predicted liking value results by calculating a weighted sum of the users' ratings, with the user similarity (to the current user) estimated by the Pearson correlation.

CF Problems

This approach can be used with items of any type; no knowledge about the domain or about characteristics of the items is needed; the only thing that matters is user ratings. The approach has some problems though:

- There is a cold start problem for new users (no user similarity can be calculated unless the new user gave some ratings).
- The cold start problem exists for new items as well (for any user, item recommendation will only work if users with high user similarity already rated the new item).
- No really meaningful explanations for recommendations can be given.
- Tuning of recommendations ("These songs are mostly good, but leave away the instrumental ones") is not possible.
- There exists a tendency to recommend the largest common denominator, i.e., recommend items that are generally well liked and known.
- Recommending items according to a specific profile is not possible.
- For users with unique taste (i.e., users for whom no users with high user similarity exist) no good recommendations can get generated.

Some of these shortcomings are addressed by the following approach.

2.6.2. Content-Based Recommendations

Content-Based Recommendations (or *Content-Based Filtering*) build on background knowledge about the items available in the database. This background knowledge typically is domain-dependent. Making recommendations to the user is a two step process: First, a profile of the items the user presumably likes is build. This can be done by various means; for example, the user can give an example item he likes, and let the profile build from that item's characteristics.

[32] `http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`

An alternative would be to let the user explicitly build a profile. Then, the approach will look for similar items; that is, items that fit the characteristics given in the profile. Similarity can be computed using various metrics such as a cosine distance metric. Of the services presented before, Pandora is a representative of the CBF approach; there, the *Music Genome Project* serves as schema, and experts create the knowledge base of item characteristics (instances).

CBF Problems

This approach is not without shortcomings:

- Building the schema for item characteristics is an additional cost.
- Gathering information about what items exhibit what characteristics is costly as well.
- Schemas are domain dependent; including a new domain is costly.
- Recommendation quality is dependent on the quality of the domain-dependent schema and the user's preferences. If the schema does not contain the characteristics the user likes, recommendations will be lacking in quality.
- Item characteristics may be of subjective nature.

CBF Benefits

However, the approach has a lot of positive sides as well:

- Explanations of the recommendations are possible.
- Fine tuning of recommendations is possible.
- Recommendation criteria can be manifold; "recommend items I might like" is only one use case. In the CBF approach, recommendation strongly overlaps with characteristics-based search. This also means that "unique tastes" of users can potentially be handled.
- While the item cold start problem exists just as in the CF approach, there is (almost) no cold start problem for new users: Simple recommendations can be generated as soon as the user gave one reference item.

Blends of the CF and CBF approaches exist; these get explained in the following.

2.6.3. Hybrid Recommendations

Both approaches presented before have different characteristics concerning requirements, features, and recommendation quality. For differing use cases and user requirements, none of the approaches is perfect [BS97, Paz99]. Therefore, combinations of both approaches, *hybrid approaches*, are a research focus;

Combining both approaches can be implemented by various means. A simple solution would be to combine the outputs of both recommenders. Another solution approach is introducing "liking" as one of the item characteristics in CBF approaches, and letting users annotate items, too, not only experts (this is what the Skipforward approach in Chapter 4 does). These and other ways of merging both approaches are discussed in [AT05]. Some approaches try to avoid focusing too much on one specific interest of the user but rather aim at giving some bandwidth in the recommendations [Sch13, SBMD11].

2.7. Tag Recommendation

One problem with annotation systems is that the Tags or concepts used to annotate resources are often numerous and might have semantics difficult to grasp at first glance. This makes the annotation process tedious and error-prone. To help with this problem, one idea is to help users when selecting relevant Tags by presenting a list of possible or likely Tags. Consequently, users are encouraged to tag more frequently, and to apply more Tags to an item, improving Recall in later searches.

CF/CBF-based Approaches

There are two major approaches for generating Tag recommendations: content-based and collaborative approaches. Content-based approaches generate Tag recommendations from the content and metadata of the resource; collaborative approaches generate Tag recommendations by using statistics concerning relations between users, resources, and their Tags.

CF Tag Recommendation

In Tag recommender approaches based on collaborative filtering, Tag recommendations exploit previous tagging behavior of of users. This includes user and item similarity concerning usage of Tags: for example, one can assume that users tend to stick to using an own specific set of Tags rather than inventing new Tags often or (mainly) reusing Tags by others; thus, Tag recommendation can base recommendations on the Tags that the current user used in the past. An item-based approach could be looking at the Tags assigned to the current item by other users, and recommending these to the current user.

CF Related Work

There exist many implementations of CF Tag recommendation. In the following, a few get listed. [JH09] explores these two approaches and integrates their output using a specific weighting scheme. [XFMS06] introduces the *HITS* algorithm that is based on CF. [JMH+07] describes *FolkRank*, a ranking algorithm based on PageRank [PBMW99]. The approach calculates a ranking of resources, users, and Tags. [LGZ08] introduces an approach that identifies topics of interests by looking at Tags, then assigns users to the identified topics, and bases Tag recommendations on these topic assignments.

CBF Tag Recommendation

In Tag recommender approaches based on content-based filtering, the item data is taken into consideration when proposing Tags. CB-based Tag recommendation shares similar advances and shortcomings of the base content-based filtering approach: Since the approach is based on information about the item and not on information entered by users, there is no cold start problem as there is in CF, and no echo chamber effect [JC09] exists. Tags that are very specific to the item in question are more likely to be recommended than in CF approaches that gravitate towards a common (albeit large) set of Tags. For direct content-based metadata (see Section 2.1), Recall and Tag recommendation Precision is potentially higher than in CF.

CBF Related Work

In the following, some examples for CBF Tag recommendation get listed. [GA08] groups Tags of the user into categories, hierarchically organizes these categories, assigns the current item to one of these categories, and then recommends Tags based on that category. [SLL+09] introduces a Tag recommender specific for BiBTeX entries, basing its recommendations on the texts listed in BiBTeX fields. [BWC07] bases its Tag recommendations on item similarity which is calculated by a document/text similarity metric and comparison of existing Tags. Thus, this approach is partially CF-based as well. [SOHB07, Mis06, CCHN07] provide similar services for blog postings and Web pages. Some approaches integrate several data sources and machine learning; for example, [LC07] uses a neural network that uses WordNet as background information to generate Tag recommendations from word frequencies and other statistics about the document that is about to get annotated. [MKS09] uses multiple information sources for generating Tag recommendations and lets users configure usage of these sources within the context of the ALOE social resource sharing platform.

To improve Recall of CB-based approaches, algorithms exist that aim at broadening the semantics of the proposed Tags, at the cost of decreased Precision. Some graph-based approaches based on Social Network Analysis [WF94] use a starting set of Tags, then extend this set. [Mic07, Beg06, SvZ08] exploit co-occurrence of Tags to achieve this goal.

Content-based Tag recommendation is not limited to text-based domains. Using different extraction algorithms, the approach can be extended to arbitrary data formats such as images, videos, or sound. For examples of approaches in these areas, see [LW08, DJLW08, BDF+03].

2.8. Ontology Wrappers

Ontology Wrappers are systems that provide formal ontologies in an ontology language (e.g., RDF(S), or OWL) from other sources that use non-machine-readable ways of representing information. In the following, related work is listed, if applicable with comparison to the *DBTropes* approach presented in Chapter 5.

Related Work

The *Linked Movie Database* [HC09] is a Linked Data source built from several other data sources such as FreeBase and DBpedia.

DBpedia [LIJ+14] has many things in common with DBTropes. In [AL07] Auer and Lehmann give details about how the semantics are extracted from Wikipedia. In comparison with DBTropes, DBpedia has a broader scope but the current DBpedia approach is inherently dump conversion, not allowing user feedback.

DBpedia Live Extraction [HSLA09] tries to keep DBpedia more up to date and improve extraction Precision using user feedback in the form of additional DBpedia specific Wiki pages, and offer a semi-automatic tool for this.

[KSR+09] describes how BBC built and published their data, and how they created linked to DBpedia resources.

[HRH08] explains publishing statistical data about the European Union (Eurostat).

SILK [VBGK09] is a "tool for discovering relationships between data items within different Linked Data sources"[33]. For the task of generating links between DBTropes and DBpedia, special matchers are necessary (for example, matching DBTropes MoviesOfThe1990s and a specific date in DBpedia), which is why using SILK was not considered for this task.

D2RQ [BC07] is "a system for accessing relational databases as virtual, read-only RDF graphs"[34]. It uses existing database data and exposes it as RDF. It does not feature extraction of data from HTML or other sources.

Comparison

None of the approaches listed above allow immediate user feedback using a built-in user interface.

[33] http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/
[34] http://d2rq.org/

2.9. Information Extraction

Information Extraction (IE) means extracting structured information from documents. Typically, this involves using methods from Natural Language Processing (NLP) to extract facts from textual documents. Among the facts that can be extracted are the following types of facts:

- *Named entities* such as people or organizations.
- *Relationships* such as personX *employedAt* organizationY; more generally, n-ary relationships are possible, adding information about time or putting the extracted fact in relation to other extracted facts.
- *Tables* from scanned images or from ASCII-only texts.

Note this list is not complete. IE is not limited to text documents. Extraction of characteristics from audio signals is possible, as is information extraction from pictures and videos.

Occurrences in this Thesis

The extracted facts can get stored along with the resource they have been extracted from in the form of annotations. In fact, this approach is pursued at several places in this thesis; in Section 3.7.3, the SCOOBIE [vEDAH09] information extraction framework was used to create annotations of documents. In Section 5.7.5, again SCOOBIE was used, this time to extract facts from text snippets in the context of a text similarity measure.

The *Annotation Recommender: Item* featured in Section 4.4.2 can be seen as a simple textual information extraction tool that uses Web search as a source for information. In a similar way, the auto-annotation functionality presented in Section 3.7.4 is a simple form of information extraction.

Related Work

The *BrainFiler* software[35] allows associating documents with categories, clustering them, and finding similar documents in a document pool. It mainly uses a TF/IDF-based approach to implement this; for each document, a term vector including term weights is built.

[Han05] is concerned with a use case similar to the one presented in Chapter 3. There, an information extraction framework (CREAM) is used to semi-automatically annotate text corpora using Semantic Web technologies. However, this approach focuses on direct content-based metadata; use metadata and other metadata the approach presented in Chapter 3 features are not covered.

[35] `http://brainbot.com/`

GATE[36] [CMBT02] is another well-known (open source) framework that is partially concerned with information extraction.

Many NLP-based systems rely on a factility called *textual entailment* that allows to draw inferences from natural language text and identify statements that can be derived from other statements [DG04]. This is similar to logic reasoning (see Section 2.2) but has a relaxed entailment definition and a strong focus on natural language whereas reasoning works on abstract representations of knowledge.

In general, IE can be seen as one of the sources for annotations that the approaches presented in this thesis store and make use of.

Summary

In this section, an overview over technologies used in this thesis has been given. Additionally, related software was presented, and differences to the approaches pursued in this thesis were highlighted. In the following, the novel approaches of this thesis are presented.

[36] `http://gate.ac.uk/`

III

Approach

CHAPTER 3
Document-Based Semantic Annotations

3.1. Motivation

The motivation behind bringing annotations to document-based applications is to enable manifold additional document- and document usage-centered functionality. This functionality potentially relies on the availability of many different kinds of additional information along with the document. Annotations enable capturing and storing information that would otherwise get lost, and thus be unavailable for additional services. To make this rather abstract notion more concrete, three specific use cases will be considered in the following.

Use Case Examples

In knowledge-intensive work, people often handle multiple sources of information in parallel, they read large amounts of text, skim texts, create summaries and overviews, (subjectively) rate information, create and organize bookmarks, or create pointers from documents to other documents.

Example use cases are:

- *Document Creation* – involving creating contracts, creating patent applications, or assessing software licenses in order to find the proper license for a new software. Characteristics are long, complex documents, with many in-document references.

- *Personal Knowledge Management* (PKM) – managing relations to people, keeping track of contacts and events, associating documents with topics, events, and people. Characteristics are short documents (*notes*), associated with desktop/Web resources, and many links between them.

- *Company Intranet Knowledge Management* – a multi-user use case involving different views on the same data: The knowledge repository contains both detailed explanation texts of projects and summaries that get built automatically from the semantic annotations available in these texts.

These use cases will also appear again in later sections. In the following, some challenges specific to these uses cases will be outlined.

Search and History

While there are tools specialized for individual domains, in general these use cases get assisted with comparatively simple means of search—desktop and Web keyword search, for example—and track-keeping features such as Web bookmarks and browser history. Keyword search and history tracking are good tools that solve many navigation and (re)discovery problems; however, for the

Challenges

tasks described above these tools are not sufficient. In the software license case, for example, keyword search is not appropriate since a set of software licenses is essentially a set of quite similar documents whose content differs mainly in meaning, not words used; unless one knows exactly what one is looking for, keyword search will not help much.

In turn, history tracking is very good for accessing information that was recently used. However, it is not well suited for looking up information that was possibly accessed a year ago. This is a major challenge when working with law texts as here, intimate knowledge of processes that are stretched over a long time is needed. Shortening the time necessarily to dive into the context appropriate to do the current work is very desirable here. This is also the case for the PKM use case, where a typical question is "What documents were relevant when I was working on project X some time ago".

These are just two examples of limitations of traditional features of document-based systems. In the following, a broader picture of the environment of document-based systems will be presented.

Document Pool

The need for better knowledge management by using meta-information can get addressed by creating a shared document pool along with meta-information in form of digests and highlightings that either get created manually or get purchased. Traditionally, this has been done using printouts and filing cabinets which have been replaced by computer-based document repositories nowadays. The issue of rediscovering previously accessed information can be handled with personal notes and digests; bookmarks partially solve this for computer-based repositories.

Wikis

In modern organizations, Wikis (see Section 2.4.1) are often an essential part of the company intranet. Such a Wiki can act as the "switchboard" for intranet information and implement the shared document repository outlined above. In the following, it is assumed that a Wiki implements the document repository.

Semantic Wikis

Normal Wikis have very limited support of any kind of annotations. The more recent Semantic Wikis (see Section 2.4.2) focus on annotation support. However, even Semantic Wikis do not support some of the traditional use cases of annotations in documents such as personal annotations (highlightings...) since they typically require to change the Wiki text to add annotations. In normal Wikis and Semantic Wikis, there exists only one view of the document and its annotations—there are no personalized views since the concept of the "cur-

rent user" is almost non-existing concerning the data model the application uses (the notion of the user exists; it is limited to keeping track of changes typically). Also, storing more elaborate metadata such as attention or user context information is not supported in normal (Semantic) Wikis, either. Even *Personal Semantic Wikis* [OVBD06] focus on knowledge formalization only—this leaves the potential of other meta-information for rediscovery and filtering features untapped.

In this chapter, the basic idea of an annotation-driven workplace in a document-centric setting is explored. The meaning of *annotations* in the scenarios is explained. A prototype implementing these ideas, *Kaukolu Wiki*, is introduced in Section 3.6.

3.2. Different Types of Annotations in Document-Centered Work

According to Wikipedia,

> "[An annotation] is metadata (e.g. a comment, explanation, presentational markup) attached to text, image, or other data. Often annotations refer to a specific part of the original data."[1]

Annotations

Annotations (see also Section 2.1) are a recurring element in document-centered work: Be it annotations in form of general comments ("This is related to xy"), ratings ("Very interesting"), or markers for further action ("Look up citation"). Annotations can be seen as a tool to individualize and personalize documents for further processing and/or later lookup. These annotations are often very specific to the individual who created them as well as specific to the situation in which the annotations have been created. This is why analyzing the meaning of such highly individualized annotations can be very hard, especially if the semantics of the annotation type is not well defined (e. g., highlighting or underlining). Using such annotations for value-adding information services is very difficult (cf. [SPMG03]).

Formal Annotations

In contrast to these rather *informal* annotations, *formal* annotations known from the Semantic Web context primarily aim at making the document's content machine-understandable. This makes the content available for automated information services (see [UCI+06, Han05] for overviews of the role of annotation in document-centric Knowledge Management and the Semantic Web).

[1] `http://en.wikipedia.org/w/index.php?title=Annotation&oldid=569893770` – accessed 23 August 2013

Most formal annotations in the Semantic Web domain can be interpreted without context.

- Semantic Web annotations are employed in the open Web scenario. Any formal annotations should be true for all users in all contexts.
- Building formal annotations is very work intensive, which means that context independence and, thus, re-usability is key to the payoff of the initial annotation effort.

This makes this approach most suitable for domains that have rather unchanging and objective knowledge as subject (see Semantic MediaWiki [KVV06] which is concerned with the encyclopedic use case).

Annotations for Workgroups

In the small working group/intranet use case, annotations often have different characteristics than in those open scenarios:

- Annotations are often created by people *not* being the author of the document that gets annotated.
- Annotations often have a very subjective character—it is perfectly possible that an annotation created by Person A is very helpful for Person A, but is not understandable for Person B.
- Often, the nature of annotations being separated from the actual document is not only due to technical limitations (such as when using a marker on paper) but a necessity even if the document exists in editable (electronical) form. Consider the original document being a finalized document with the person reading it wanting to add personal notes to it.

As already established before, for effective and personalized knowledge management, PKM software should support these types of annotations in document-centered systems.

Annotation Types

In detail, the following types of annotations are of special interest:

Formalization uses annotations that describe the semantics of the annotated text, making the text's information available in a machine-readable form (such as RDF/S, see Section 2.2). This is the type of annotations typically used in open Web scenarios as well, and the most common type of annotation in the field of Semantic Wikis.

Conceptual annotations are used to classify/tag document passages. To provide machine-readable semantics for this classification, an ontology is needed. At DFKI, the Personal Information Model (Ontology) (PIMO) is used, i.e.,

the user(s)'s conceptualization of his/their (knowledge work) world, see Section 2.3.1. Instead of tagging passages with bare text labels, PIMO concepts are used to annotate and classify the passages. This allows sophisticated retrieval since connections of the concepts used are available to the software.

Attention annotations carry information about how much attention each part of a document got by the user. The information source for creating this kind of annotations could be driven by different technologies such as eye trackers or scrolling-based software observers, see Section 2.3.3.

Highlightings and comments typically get created manually and are straightforward annotations. Viewed on their own and without contextual information, these annotations carry limited value, at least concerning automated processing, since it is very difficult to assign or elicit semantics from them. However, by combining these annotations with contextual information, more insights into the user's intentions when creating the annotation can be gained.

Contextualized annotations are a feature that allows interpreting annotations in the context of their creation. By enriching annotations at their creation time with information about the context of the user (see Section 2.3.2), personalized and context-dependent views of the (annotations of) a document become possible. This includes features such as specifically highlighting parts of the document that a certain user read while working on a specific contract. User context makes use of the aforementioned PIMO as well.

In the following, storing and handling these kinds of annotations will be explained in more detail.

3.3. The Wiki Workbench

The Wiki workbench is the key system component for handling annotations. In the following, an overview over the Semantic Wiki idea, about the typical features of a Semantic Wiki, and what extensions are needed to realize the vision stated in the motivation of this chapter will be given. This is partially based on [vEKS+08].

Annotation Usage in Normal Semantic Wikis

Semantic Wiki Definition

According to Wikipedia,

> "A semantic wiki is a wiki that has an underlying model of the knowledge described in its pages. Regular, or syntactic, wikis have structured text and untyped hyperlinks. Semantic wikis, on the other hand, provide the ability to capture or identify information about the data within pages, and the relationships between pages, in ways that can be queried or exported like a database."[2]

Typically, Semantic Wikis implement this by associating Wiki pages with semantic resources, and allowing links between pages to get typed (see also Section 2.4.2). Types can usually be set according to an ontology available to the Wiki.

Example Annotation

An example for annotations in most Semantic Wikis is shown in Figure 3.1. The annotation is shown only conceptually. Ontologies used are not shown explicitly for brevity. In the example, an excerpt of the Wikipedia Wiki page about the city "Dublin"[3] is shown. In a Semantic Wiki, this page can get tagged as an instance of City, with City being a page representing the class of cities. In the text of the page, Ireland is referred. The link the Wiki displays when rendering the Dublin page may then get typed with *lies_in*. Alternatively, some Wikis allow form-based annotation of pages: If a page is tagged with the type City, a form may get generated on that page, allowing to enter (missing) properties for this City instance. The data collected by these means is typically added to the Wiki markup using an extended Wiki markup syntax.

Wiki implementations differ in the way annotations are implemented in the Wiki markup (see Section 2.4.2 for example markup). They also use several ways of displaying them such as mouseover tooltips or explanation boxes next to the actual article.

Drawbacks

However, while this approach is elegant in terms of simplicity and ease of use, there are several drawbacks:

- The rigid mapping between Wiki pages and semantic resources imposes severe limits on the possible use cases. The knowledge present in a large typical Wiki page would correspond to a resource with hundreds of properties—or possibly many resources and many associated triples. To give an example, say a Wiki page containing a table listing cities and

[2] http://en.wikipedia.org/w/index.php?title=Semantic_wiki&oldid=561796523 – accessed 27 June 2013

[3] http://en.wikipedia.org/w/index.php?title=Dublin&oldid=176993469

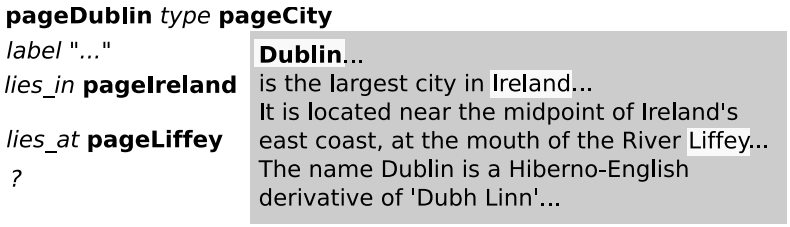


Figure 3.1.: A Wiki page about *Dublin* as annotated in many Semantic Wikis [vEKS+08].

their population should be mapped to one resource *per city* plus further describing triples. With a rigid one resource *per page* mapping, this mapping is impossible since all triples originating on one page have to have the same subject.

- Handling of existing documents, be it existing Wiki pages or other documents, is difficult. Metadata has to be added into the page text, changing the actual document. For texts such as law documents or finalized versions of documents this might not be desirable.
- Handling of further information concerning annotations such as provenance or context information is difficult. Personal annotations are not supported.

In the following, an approach to solve these challenges is described.

Documents, Annotations, and Links Between Them

For most of the types of annotations described before, being able to annotate any part of a document (or here, Wiki page) is necessary—be it a single word, a part of a sentence, a paragraph, or the whole document. It is evident that this kind of annotations cannot be represented in extended Wiki markup, at least not in a user-friendly way. Thus, the focus in the following is on annotations that are *not* represented in Wiki markup.

Text Annotations

In the following, the annotations described are similar to annotations or *notes* created in a standard word processing application. These are displayed in connection with the text they are associated with but do not show up as text characters or markup in neither editing nor viewing mode unless requested by the user.

With this kind of annotation implementation in mind, the example presented in Figure 3.1 can be approached in a different way. In Figure 3.2, the whole doc-

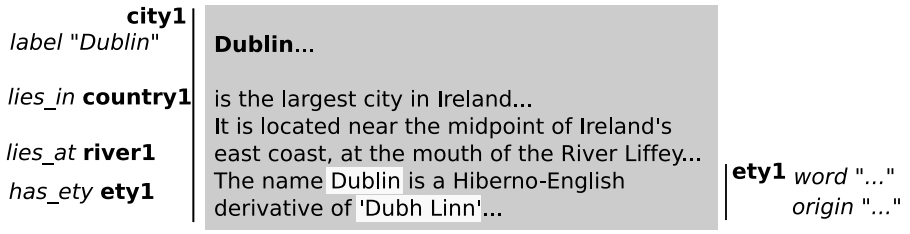


Figure 3.2.: Annotated *Dublin* page [vEKS+08].

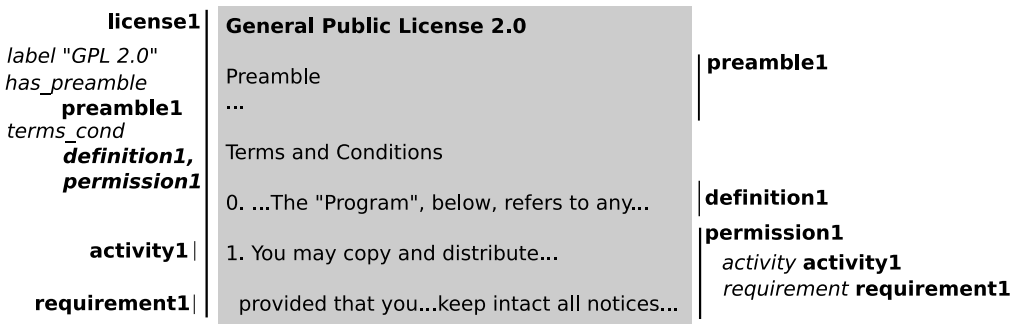


Figure 3.3.: Annotating a software license [vEKS+08].

ument has been annotated with the Dublin instance of a City. "Ireland" and "Liffey" are textual occurrences of the corresponding Country and River instances.

So far, this is not to be very different from the standard Semantic Wiki approach. Using this approach, however, it *is* possible to annotate the etymology information here, too: The textual statement describing provenance of the name "Dublin" has been annotated with an instance of an EtymologyFact class in a (for the sake of this example fictional) etymology ontology. This instance holds further describing information as shown in the picture.

Complex Annotations

Another example can be seen in Figure 3.3 that depicts a software license text and its annotations. This text contains lots of separate mentions of certain facts–after all, a license is a collection of legal statements. It is desirable to have a formalized description of the structure of the text. However, using the page-resource mapping technique, there is neither the concept of a *paragraph* of a page available nor is it possible to point to a paragraph: Expressing *structural metadata* (cf. Section 2.1) is not possible in that approach. Only a kind of simple tagging would be possible–annotating a license with a few rather generic information fragments, de facto using an ontology with one or few classes and lots of properties, and creating one large RDF instance.

Using the text annotation approach, fine-grained annotation is possible. Both

text decomposition and assigning complex fact representations to individual text fragments can be done. Since a tight connection between annotations (and the information contained therein) and the text exists, retrieval of text passages that concern or express certain facts can be done quite easily: The smallest unit that can be queried for is one annotation (i.e., a selection of text that may be as small as one character).

3.4. Annotation Origins

Using the annotation model outlined above, handling numerous novel sources of annotations becomes possible. These sources can broadly be categorized by whether their data can be gathered automatically or manually.

3.4.1. Manually Created Annotations

For manually created annotations in the form of embedded Wiki markup, any RDF facts included in the new markup are made available for RDF queries.

Text annotations associate text with RDF instances of ontology classes: for example, one can annotate a text occurrence of a company with an RDF instance of the RDFS class Company. The properties of the RDF instance (such as *established_in_year* for the Company example) can be filled typically using a form that gets presented to the user when creating the annotation.

Annotation Recommender

Cascaded Annotations

For complex documents such as software license texts, *cascaded annotations* can be used to represent the document's logical structure (cf. *Structural Metadata* in Section 2.1) and internal references. This allows formally describing facts such as *This section consists of a set of Requirements* or *This is a Constraint limiting the rights granted in that other section*. This involves annotations that reference other annotations, thus the name *cascaded annotations*.

Since the structure of the data contained in annotation instances is defined by their RDFS classes, it is possible to build assistance services for creating these annotation instances. Two ways of assisting the user when creating cascading annotations come immediately to mind:

- Filling dangling fields in annotation instances—e.g., if a text has been annotated with a SetOfStatements annotation instance that can point to any

Statement, it is recommended to present (subtypes of) the Statement annotation type to the user for creating a new annotation (along with any existing Statement for reference).

- Exploiting textual location—i.e., if the user wants to create a new annotation in a text part located in a bigger text section that already has been annotated with a SetOfStatements instance, proceed similar to the case before: present (subtypes of) the Statement annotation type along with any existing Statement.

3.4.2. Automatically Gathered Annotation Data

Some sources for annotation information that stems from the user and work/usage context can get tapped.

Attention Information

Information about the amount of attention a user has given to the documents in the document pool can be gathered just by analyzing the usage behavior (*What documents have been retrieved and shown on user request? What search results has the user clicked on?*). Even attention given to *part of* a document can be measured (indirectly) by observing usage such as scrolling behavior on the screen, or using a more heavyweight approach such as an eyetracker.

Attention Annotations

This additional usage information can be persisted as special attention annotations such as Read or Skimmed annotations denoting whether a user has read or skimmed (watched briefly) a text passage.

These annotations are helpful for later retrieval tasks, most notably re-finding tasks (*What passages did I read in this document last week?*).

User Context

User context is information such as *What project is the user currently working on* or *What topic is most important in the current task*. This information can be elicited from user interaction with the computer by various means (see Section 2.3.3) and is available at all times once the user context system has been set up. While there is little sense creating pure user context annotations (i.e., attaching user context information to a part of a document without any further semantics),

Annotation Metadata

use can be made of this information in other ways: Treating user context as additional annotation metadata is beneficial. User context can act as another situational descriptor similar to the time of creating an annotation or the user

that created an annotation: *This annotation has been created in the context of project XY*—and probably has some specific significance in the context of that project.

This additional annotation metadata becomes especially important when combined with attention information. Knowledge of what document (parts) are especially relevant (since they have been read) in the context of project XY can be used in later retrieval and summarizing actions.

3.5. Uses of Annotations

3.5.1. Annotation-Driven Search

One of the key features that annotations enable is annotation-driven search. All data and metadata found in the system can be queried. Combinations of different query types should be possible: Queries such as *Show me text paragraphs I read in the last year in the context of project XY* should be possible as well as queries like *What paragraphs are referencing software license YZ* or *What documents containing the word "open source" have references to what software products?*. These queries combine search for attention annotations, attention metadata about context, search for annotations of a specific type, and plain text search.

Example Queries

3.5.2. Personalized Views

Using personal annotations, building personalized views of documents becomes possible: for example, passages that have received a lot of user attention in the past can get highlighted, and passages that carry annotations stemming from a specific user context can get marked. Both features help sifting through large documents and finding passages that are relevant in the current task.

3.5.3. Ontology-Structured Views and Navigation

The data carried in the sum of all annotations represents a knowledge base: Each annotation is an instance of an RDF class, carrying information about the concept it represents: for example, if the documents in the Wiki are about companies and their employees, and the documents are properly annotated with representational annotations, there will be many annotation instances of people/employees, each associated to the company the person works at. Using a visualization of the RDF data, the knowledge worker can browse it. Advanced features such as SPARQL queries (see Section 2.2) on the RDF data can be implemented, as can be template-based RDF rendering. Moving from the docu-

ment+annotations view to RDF view is possible, as is the navigation back from RDF view to document view.

3.6. Prototype Implementation: Kaukolu Wiki

In this section, an implementation of the Semantic Wiki component outlined in Chapter 3 is presented. This implementation is called *Kaukolu*.

3.6.1. Design Choices

The design choices concerning the underlying annotation model have been discussed in the previous section. In the following, a number of additional design choices are presented. These have their roots in partially non-functional requirements, and the need to provide manageable user interfaces to a complex system.

Extend Existing Software

One important decision when developing Kaukolu was that the focus of development efforts was to be put into developing actual new semantic features. Thus, developing a Wiki from scratch was not an option. Since the department's programming language of choice at the point of inception of Kaukolu was Java, a survey of Java-based Wiki implementations was done. In the end, JSPWiki[4] was chosen as basis for Kaukolu; it was reasonably well organized, had a fitting feature set, and last but not least a lively community. Complying with JSPWiki's LGPL open source licensing, Kaukolu was made available as open source as well[5]. Other components and libraries have been added to Kaukolu in the course of the development; among other things, it uses Sesame[6] as RDF triple store and Dojo[7], a JavaScript framework, for its interactive user interfaces.

Other Libraries

Multi-Purpose Tool

Kaukolu has been designed to be versatile concerning RDF: Any RDF can be imported and edited whereas other Semantic Wiki implementations mostly impose restrictions on the RDF that can be handled or use RDF as export format only. While Kaukolu was not supposed to act as an ontology editor primarily (this was out of focus of the Semantic Wiki idea and too tall of an order anyways), it should at least be possible to somehow edit the RDF(S) stored in and used by the Wiki. Additionally, there was the requirement to be able to import and export arbitrary RDF and to allow statements about arbitrary RDF subjects on any Wiki page (see Section 3.3). This need arose due to considerations of practicability: Turnaround times for exporting annotation ontologies to

Pragmatic RDF Editing

[4] http://jspwiki.apache.org/
[5] http://kaukoluwiki.opendfki.de/
[6] http://www.openrdf.org/
[7] http://dojotoolkit.org/

an external ontology editor, changing the ontology as needed, and re-importing would have been a major obstacle in working with structured annotations. This lead to the idea to support a simple RDF syntax in the Wiki. A direct benefit of this it that users are able to edit and extend the ontologies used by the Wiki in a straightforward way, using all features a Wiki provides (versioning, collaborative authoring, viewing diffs, ...).

Detached Annotations

The main annotation model pursued in Kaukolu is the detached variant: Users can select text parts and create annotations carrying metadata for these text parts. This is in contrast to most existing Semantic Wikis which rely on extended Wiki markup (c.f. Section 3.3). Apart from higher flexibility concerning annotation expressivity and interfacing with external information such as usage information, this facilitates simplified user interfaces. Extended markup comes with a steep learning curve; even advanced support when entering markup (auto-complete and other context-sensitive help) cannot change the fact that the user essentially is editing source code. Detached annotations, instead, allow creating annotations using practices that are known to the user such as marking a passage, selecting a type of annotation, and filling out the required information in a form.

Semantic Search

For semantic search, a major design choice was to limit search to text paragraphs as results. This allowed to keep the search interface clean. Being able to not only search for (annotated) text paragraphs but for annotations themselves, or arbitrary RDF, was tempting, but would have cluttered the user interface, and probably confused the user—people are not used to have totally different types of search results in the same user interface. Besides, there are other places in Kaukolu that allow queries on annotations and arbitrary RDF in a more natural way, such as inline queries in Wiki pages that build tables from annotation metadata (e.g., person or project lists).

Personalized Views

For personalized views, a simple drop down-based user interface was chosen, implementing filtering of the annotations shown. An exception was made for attention information; since this tends to span the whole page, attention annotations are typically not displayed as normal annotations (i.e., text highlightings with mouseovers). Instead, based on user choice, attention information is ignored, or text is shaded/hidden according to attention annotations. The interface design for this functionality was built (i) following requirements in example projects, (ii) emulating similar legacy applications (for the attention tracker part), and (iii) incorporating user feedback during project evaluations.

In the following, a walkthrough of the Kaukolu system is given. This details functionality implemented by the system as well as the specific user interfaces implementing this functionality. Finally, in Section 3.7 a number of use cases building on this functionality is presented.

3.6.2. System Outline

The list below gives a quick overview of the features that have been implemented in Kaukolu. Most of these features are presented in context with different use cases in Section 3.7. The more technical features are explained on the Kaukolu homepage. After each block of features, a rationale is given for the need of the enumerated features.

- **annotation support**
  - all annotations are based on RDF/S
  - support for annotations separated from content (*detached annotations*: "highlighted text" that is linked to an RDF annotation instance)
    - annotation info can be entered using forms
    - support for annotations that reference other annotations (*cascaded annotations*)
    - recommender for annotation types if using cascaded annotations
  - special markup for arbitrary RDF is supported: allows importing ontologies and editing them in a simple way
    - autocompletion support when authoring RDF markup
    - alias support for shortening URIs
  - support for special annotations such as drawing annotations

Basing annotations on the *RDF/S standard* was necessary to be able to interface with other Semantic Web projects and reuse ontologies. With a proprietary annotation data model, many of the use cases presented later would not have been possible.
Detached annotations are the enabling technology for supporting creating annotations automatically. For example, user attention information would not have been feasible to implement if creating annotations changed the actual Wiki markup. As the aim of Kaukolu is to act as a switchboard for document annotations, supporting a broad number of sources of annotation information is crucial. Detached annotations are a feature that sets Kaukolu apart from most other Semantic Wikis.
RDF-enabled markup complements detached annotations. The need for this feature arose from the Semantic Desktop use case. There, the Wiki was often used for annotating Semantic Desktop resources with arbitrary free text. In these texts, often information that was easy to formalize was present ("The person

[this text is about] attended the NEPOMUK kick-off."). RDF-enabled markup allows both to write this information faster (thanks to semantic autocompletion support), and make the information available in a machine-readable form. *Drawing annotations* were a natural requirement from the Mymory tablet PC use case.

- **annotation metadata support**
  - all annotations carry metadata about their creator and creation time
  - additional metadata such as creation context, etc., is supported as well
- **attention capturing** keeps track of user attention given to (parts of) a document
  - attention annotations may get enriched with creation context
  - software/scrolling-based or based on special hardware (eyetracker)

These features were integral in the Mymory use case that mostly focused on long documents. Without attention capturing, re-finding information and relevance weighting in long documents would be difficult. Additionally, user attention is one source of metadata illustrating the need for supporting detached annotations. The ability to process user attention information is unique to Kaukolu in the Semantic Wiki domain.

- **structured data visualization and navigation support**
  - all data available as RDF (annotations and their metadata, ontologies, etc.) can be visualized as plain RDF or using templates
  - navigation from Wiki page view to structured view and back is supported

With much of the information in the Wiki texts available in a machine-readable form, building summaries from this information automatically becomes possible. Without machine-interpretable data, summary pages and extra navigational help in Wikis need to be built manually, i.e., as Wiki markup containing lists and links between Wiki pages. With the data available as RDF, these summary pages can get auto-created. This is a feature found in many Semantic Wikis.

- **annotation-aware search**
  - faceted paragraph search allows looking for...
    - annotations with a specified type
    - annotations that carry specific data
    - paragraphs containing search text
    - paragraphs a specific user read
    - annotations created by specific people
    - annotations created in a certain user context
    - etc...
  - creating new Wiki pages from search results

Detached annotations allow fine-grained paragraph search. This type of search is the implementation of attention-driven information retrieval and a precursor technology for building personalized views. Paragraph search of this type is unique to Kaukolu with regard to the domain of Semantic Wikis.

- **annotation-aware viewing**
  - annotation filtering in Wiki page view mode
    - by user, creation context, time, etc.
  - filtering Wiki page text by attention annotations
  - inline queries on background RDF/annotation data

In effect, *annotation-aware viewing* is the continuation of applying annotation-driven paragraph search for building personalized views. It is less cumbersome than annotation-aware search and can be easily enabled while just browsing Wiki pages. For the Mymory use case stressing long documents, this functionality was a requirement.

- **non-semantic features**
  - PDF import
  - extended HTML import (including annotations)
  - microcontent: include content from other pages, optionally in collapsed/expandable form
  - versioned delete/rename of pages

  - LDAP authentication interface
  - advanced mail change notifications using ripple down rules

These features arose from different requirements: The *PDF/HTML import functionality* was necessary to import large amounts of texts into Kaukolu. Especially in the Mymory use case, copying texts by hand would have been very tedious otherwise; also, formatting would have gotten lost this way. In the iGreen use case PDF import proved useful as well.
Versioned deletion/renaming and *LDAP authentication* is a requirement that stems from the deployment of Kaukolu within the DFKI KM department in an "eat your own dogfood" approach. There, refactoring of Wiki contents are frequently done (then, being able to track page deletions and renames is useful); LDAP authentication is useful in order to allow users to login with their company credentials, in contrast to requiring users to create new accounts for just Kaukolu.
Mail notifications enabled several internal use cases. To give an example, using notifications people could subscribe to a Wiki page concerning orders at an electronics store. While that page would often not see updates for a long time, as soon as somebody wrote "will order tomorrow" on the page, people would get notified automatically, and could add items they wanted to order as well.

Connection with Scenarios

In the following the scenarios presented in Section 3.1 are listed, highlighting the features that are especially relevant in each context.

- *Document Creation* – characteristics were long, complex documents, with many in-document references. Here, detached annotations are of high importance, since changing documents just to add annotations is not an option. *Cascaded annotations* are useful as well, as they are instrumental for describing the structure of the complex documents involved in this use case. *Attention annotations*, combined with *annotation context* metadata, are interesting in case some documents are used in different contexts (e.g., one software license is of interest in several projects). Several of the new technical features such as PDF import are also very helpful. A detailed example is shown in Section 3.7.1.

- *Personal Knowledge Management* (PKM) – characteristics were short documents (*notes*), associated with desktop/Web resources, and many links between them. Here, *autocompletion support* is very helpful, as it facilitates quick linking between documents and resources. *Structured data visualization* helps for getting an overview of the knowledge network. A detailed example is shown in Section 3.7.2.

- *Company Intranet Knowledge Management* – here, existing data can be reused by importing RDF data and visualizing it using *structured data visualization*. Additional documents can get imported using the PDF importer, and possibly automatically get annotated at the same time, by an external annotation component. An example for this is described in Section 3.7.3. The Wiki has also been in use at DFKI internally; there, the LDAP integration and advanced mail notifications were very helpful.

3.6.3. Implementation of Semantic Annotations

Detached Annotations

In Section 3.3, the benefits of annotations supporting arbitrary subjects and of annotations that are stored separated from the actual document have been shown. This annotation model is one of the annotation models supported by Kaukolu.

Instances and Anchors

Technically, whenever an annotation is created, Kaukolu creates an RDF instance of the RDFS class of the annotation in question. This class has to be a subclass of the Annotation RDFS class that is foundational to Kaukolu. In order to associate the instance with Wiki text, Kaukolu additionally creates an AnnotationAnchor which is an internal helper class and typically not directly visible to the user. This anchor associates the RDF resource of the annotation with a part of the Wiki markup by storing character offsets of the annotated text. If the markup gets edited (and, thus, offsets change), offsets get updated by a modified text diff algorithm.

User Perspective

From user perspective, creating manual annotations in Kaukolu is pretty straightforward. Once a text part to be annotated is selected, right-clicking opens an annotation window where possible annotation types are displayed. These types and corresponding dialogs are fetched from ontologies loaded in Kaukolu's RDF repository or, if configured accordingly, also from external sources using a custom implementation.

Since typically the system knows many possible annotation classes, there are some shortcuts for potential interesting types, exploiting inherent annotations characteristics: (i) If the text that is to be annotated lies within a text that has already been annotated, RDFS ranges of properties of the surrounding annotation instance are shown. This facilitates text decomposition: for example, if a text segment is annotated as being a CollectionOfRequirements (with this class naturally having a *hasRequirement* property with Requirement as range), then this is taken as a hint to display Requirement (instances) more prominently. (ii) Another way of limiting the class tree shown is looking at the history of recently

used classes and showing the class hierarchy in their vicinity only.

User Context

Whenever an annotation is created, additional user context including most relevant PIMO concepts and other metadata such as author and creation time is attached.

If a paragraph contains annotations, typically icons representing the annotation types are displayed next to the paragraph in viewing mode. If the mouse is moved over the paragraph or icons, the annotated text section is highlighted and a popup containing information about the annotation is shown (Figure 3.4).

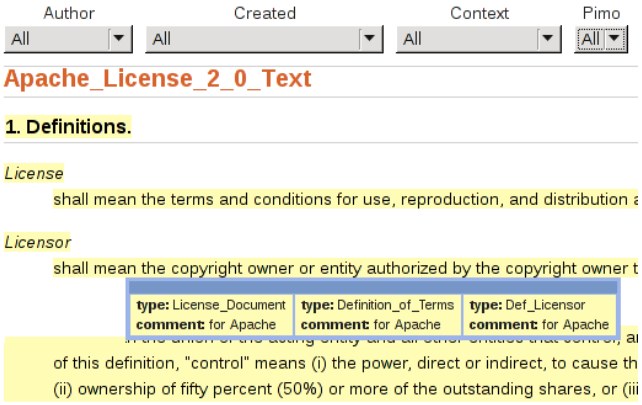


Figure 3.4.: Multiple overlapping annotations.

Semantic Markup and Autocompletion

For simple management of arbitrary RDF, Kaukolu supports a markup syntax similar to N3. It also supports *aliases* which act as shortcuts to URIs and make entering structured data easier. However, even with special syntax and aliases for properties and instances, entering RDF triples is a tedious task. Without further support, the user would need to keep the documentation of the ontologies always at hand, typing mistakes would introduce severe errors, and the user would have to remember the URIs of all RDF instances created in the Wiki.

Ontology-Based Autocompletion

In Kaukolu , there is ontology–based autocompletion support, which proposes aliases based on RDFS range and domains: for example, when typing *Paul knows*, with *Paul* being an alias for a foaf:Person, and *knows* being associated to *foaf:knows*, the system automatically proposes a list of foaf:Person resources available in the Wiki to complete the RDF triple, as only resources of the type foaf:Person are allowed as range of *foaf:knows*, even without any prefix typed. If a prefix has been typed already, it is used to narrow down the list of suggestions. Autocompletion works for predicates, too. In case no alias is found in the

typed text, Kaukolu assumes that the user does not intend to write triples, and simply proposes names of Wiki pages as autocompletion suggestions, based on the prefix already typed. If the user types "InfoOn", and there are "InfoOnPaul" and "InfoOnSarah" pages in the Wiki, those both page names are suggested.

Semantic markup is mostly used when importing arbitrary RDF, for example ontologies created in external editors. It has also been used extensively in the NEPOMUK project (see Section 3.7.2).

3.6.4. Annotation-Driven Search

All data and metadata found in the system can be queried in Kaukolu's semantic search feature. Search always returns Wiki text paragraphs as results; searching for standalone RDF resources or authors directly is not supported. This was done to keep the system simple and to keep some resemblance with a normal Wiki search in which users expect text passages to be returned. In the backend, so-called *filter* components search paragraph (identifiers) that match the selected criteria. Every filter can use a different data source for its search. In Kaukolu, three filters have been implemented:

Filters

- *A page filter* supporting standard Wiki search (full text page content search, search by author, search by modification date).
- *An annotation filter* searching for paragraphs with a matching Kaukolu annotation. Annotations can be filtered using facets derived directly from their RDF representation which is useful when handling annotations based on arbitrary ontologies. Filtering by the annotation's author and creation date is possible, too.
- *A context filter* implementing filtering by an annotation's context. This is basically a shortcut to improve searching for annotations created in a special context which is tedious (but possible) to do using the annotation filter.

Combining Filters

Filters can be combined using AND and OR operators. Once a filter is selected, the user can choose from a number of facets supported by the filter: for example, for the page filter, the user can choose between page content (text), page author, and modification date. Then, a restriction value for this facet can be chosen. If the range of the facet is discrete, only values that exist in the Wiki are displayed. For the annotation filter, facets and restriction values are derived from actual RDF annotations in the Wiki.

Shortcuts

For common use cases, a shortcut exists that lets the user enter plain text for a filter; the filter then shows a list of restrictions the user can choose from. Typically, the filter will match against typical facets and their valid restriction values and display possible combinations: for example, a page filter given a shortcut string will match against page content and author.

An example for annotation-driven search can be seen in Figure 3.5. There, the user searched for an annotation instance of the type Permission that references (via the property *activity*) an annotation instance of the type Copy, or in other words, a part of a license text that is concerned with permissions to copying the work.

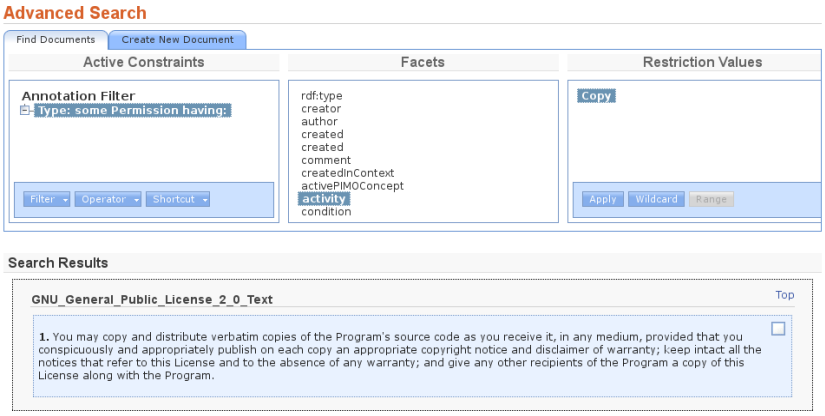


Figure 3.5.: Searching for annotated paragraphs.

Creating Documents Based on Search Results

Text passages found in search can be used to create new documents. The idea here is that this way it is possible to "remix" texts to form documents that fit to new requirements. Passages are copied, not referenced—however, provenance information indicating the source of copied paragraphs is kept as a new annotation.

3.6.5. Personalized Views using Annotations

Attention Handling

Kaukolu supports filtering pages based on annotations during normal page view. Two major ways of filtering have been implemented. One can filter Wiki pages by attention information— in practice, this means that passages bearing a *read* or *skimmed* attention annotation (see Section 3.6.7) are displayed as usual

Context Handling

while all other passages are grayed out. This helps finding relevant passages in large documents. Filtering by context information is implemented differently—as context information is meta-information about annotations, we filter the annotations to display here. For example, one can choose to display annotations created only in the context of a certain project or topic using a drop-down menu.

3.6.6. Ontology-Structured Views and Navigation

The idea of ontology-structured views and navigation has already been outlined in Section 3.5.3. Kaukolu supports rendering of RDF primarily using template approach. If the user navigates to an annotation or generic RDF resource that has a type for that a template exists, that template is used for rendering. Figure 3.6 shows an example of the view of such a resource. If no template is available, Kaukolu falls back to generic triple-based rendering. Templates support a multitude of features known from standard templating languages; a special feature is performing SPARQL queries and rendering the results. Users can also navigate from resource to resource in this view, or switch back to document view, as document/passages referencing the currently viewed resource are listed.

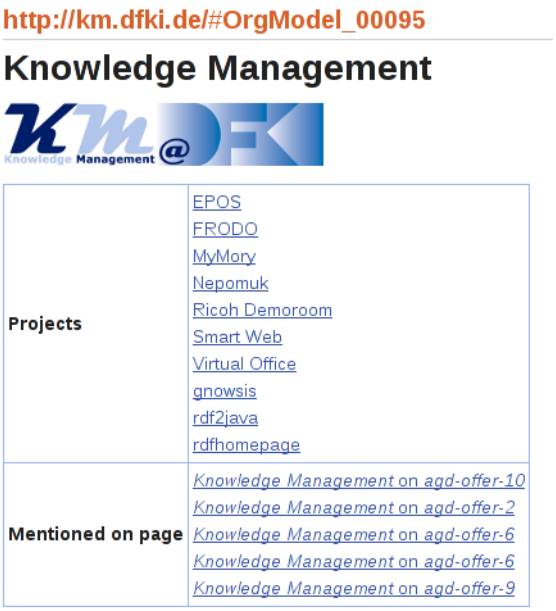


Figure 3.6.: Template-based rendering of RDF resources and annotation data.

3.6.7. Integration of Annotation Sources

Kaukolu allows to store annotations from a number of sources. In the following, integration of some sources is outlined.

User Context

Whenever an annotation is created (be it automatically or manually), the current user context is attached to the annotation. This can be seen in Figure 3.7; there, the RDF data representing an annotation is shown. This annotation has been supplied with context information; apart from a specific context URI (last line), individual PIMO concepts are stored as well, for easier search. Here, the concepts for the people "Malte Kiesel" and "Sven Schwarz" are visible as well as the projects "Mymory" and "NEPOMUK".

Referencing Pages

» Apache_License_2_0_Text

All Occurrences

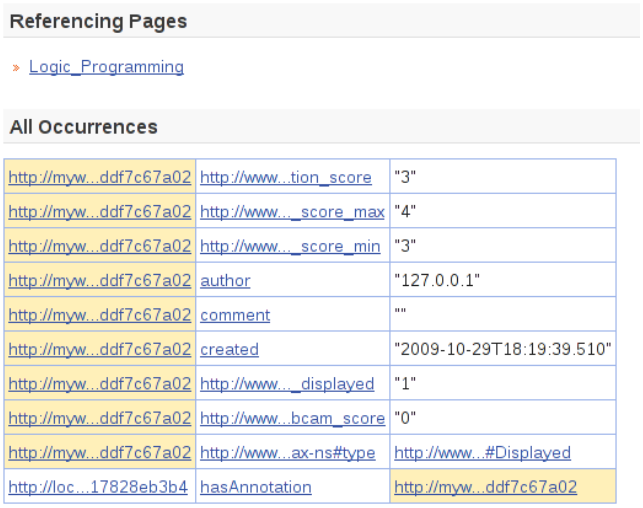
http://myw...34e590ec5f	http://www...IMOConcept	gnowsis://...lte+Kiesel
http://myw...34e590ec5f	http://www...IMOConcept	gnowsis://...imo/Mymory
http://myw...34e590ec5f	http://www...IMOConcept	gnowsis://...mo/NEPOMUK
http://myw...34e590ec5f	http://www...IMOConcept	gnowsis://...venSchwarz
http://myw...34e590ec5f	author	"Anonymous"
http://myw...34e590ec5f	comment	"for Apache"
http://myw...34e590ec5f	created	"2008-02-13T13:08:55.569"
http://myw...34e590ec5f	http://www...dInContext	urn:contex...1T17.22.41

Figure 3.7.: RDF data for an annotation, showing PIMO concepts.

User context can also be used for building personalized views; the annotations shown while browsing can be limited to those from a specific context or those with a context that features a specific PIMO concept. The chooser for these PIMO concepts can be seen in Figure 3.4 at the top right corner.

Attention Information

In Figure 3.8, the technical details behind an attention annotation as described in Section 3.4 are shown. In document view mode, this information can be used to gray out text parts that only have been skimmed. This allows quickly finding text parts that have gotten lots of attention in the past.



Referencing Pages

» Logic_Programming

All Occurrences

http://myw...ddf7c67a02	http://www...tion_score	"3"
http://myw...ddf7c67a02	http://www..._score_max	"4"
http://myw...ddf7c67a02	http://www..._score_min	"3"
http://myw...ddf7c67a02	author	"127.0.0.1"
http://myw...ddf7c67a02	comment	""
http://myw...ddf7c67a02	created	"2009-10-29T18:19:39.510"
http://myw...ddf7c67a02	http://www..._displayed	"1"
http://myw...ddf7c67a02	http://www...bcam_score	"0"
http://myw...ddf7c67a02	http://www...ax-ns#type	http://www...#Displayed
http://loc...17828eb3b4	hasAnnotation	http://myw...ddf7c67a02

Figure 3.8.: RDF data for an attention annotation.

3.6.8. Annotation Recommendation

Form-Based Annotating

The idea of document/structure-related annotation recommendations has been outlined in Section 3.4. Kaukolu implements this using functionality included in the form-based annotation creation process. In Figure 3.9, the form that is used for creating annotations is shown. On the right side, it can be seen that an instance of the currently selected Definition_of_Terms annotation class allows to reference, among other things, Statement annotations, which is specified in the underlying annotation ontology. Therefore, only annotations of the Statement type are shown in the drop-down in the form.

Overlapping Annotations

Another recommendation feature is the "overlapping" tab seen at the top of the left. In that tab, classes are shown that may get referenced by any cascading annotations that surround the currently selected text part that is to be annotated. This allows workflows targeted at structural annotations—for example, a section of the document that represents a collection of statements can be annotated with the corresponding class. If the user starts to annotate a part of text that lies within the area annotated with Collection_of_Statements, the "overlapping" tab will show, among other things, the Statement annotation class.

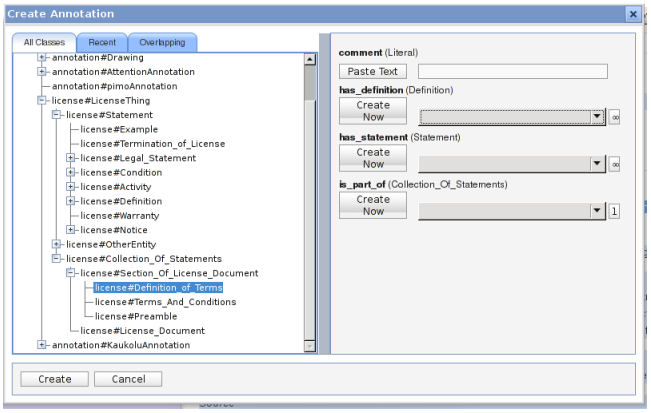


Figure 3.9.: Form-based annotation creation including context-specific recommendations.

3.7. System Usage Examples

In the following, an overview of projects Kaukolu has been involved in will be given. This demonstrates its features and capabilities as well. It also highlights Kaukolu's versatility for different use cases.

3.7.1. Mymory

According to its Website[8]:

> The vision of Mymory is
>
> - to employ technologies for unobtrusive user observation in order to create relations between information items that are meaningful to the user in his specific context,
> - to use attention evidence for more precise information delivery, and
> - to provide mechanisms of Meaning Coordination to facilitate reusability of knowledge among different contexts.
>
> Ultimately, Mymory will lead to a personal memory for knowledge workers which is not only a passive storage, but also proactively supports context-driven structuring of its content and user-perspective interpretation and incorporation of arriving information.

[8] http://www.dfki.de/mymory/

> Results of the various project lines will be demonstrated within the C3DW Application: The Connected, Context-aware, Creative Document Workspace which will be embedded in an enriched physical desktop environment.

C3DW Application

Kaukolu is a central part of the aforementioned C3DW application, representing document repository and switchboard for all annotation information the Mymory system including its context and attention components generate.

Mymory Ontologies

The main Mymory scenario is based on a set of software license texts, project Websites, and Wikipedia articles that are managed and annotated in Kaukolu/C3DW. Annotations are based on a software license ontology (that allows formalization of software license semantics), user PIMOs (containing representations of projects and people, used for modeling contexts), and a generic annotation ontology (facilitating simple annotations such as rating texts or highlighting). Parts of these ontologies are visible in Figure 3.5 and Figure 3.9.

Figure 3.10.: The standard Mymory workplace setup.

In Figure 3.10, the Mymory workplace setup is shown. On the screen on the left, a search application is shown. In the middle, Kaukolu is visible, running on a screen that features a built-in eyetracker (seen in detail in Figure 3.11). On the right screen, the user context interface is shown.

The search application allows looking through the text documents available in the Mymory system quickly using fulltext search with weighted terms. The user context interface shows a selection of contexts the user is known to pursue. Each context is characterized by a number of PIMO concepts with specific activation values. In the context user interface, the user's current context is shown. It also allows setting the context explicitly in case the automatic context detection component fails to detect a user context switch or selects the wrong context.

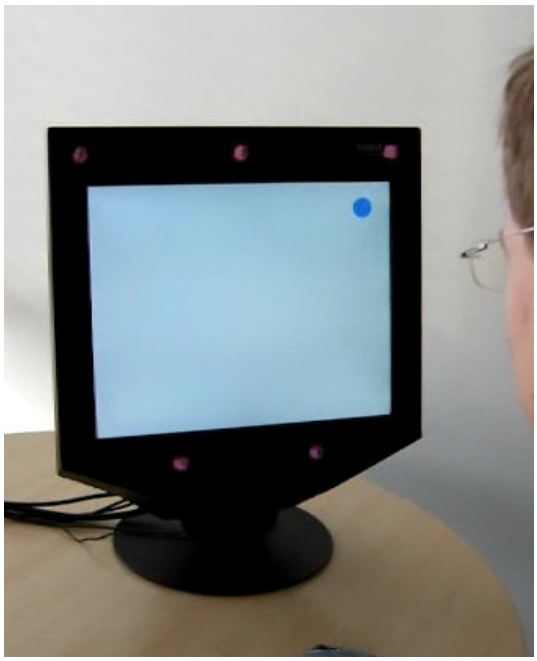


Figure 3.11.: Calibration of the eyetracker (infrared lights are invisible to the human eye).

The eyetracker (Figure 3.11) allows detecting the point the user is looking at with an accuracy of very few centimeters. It is connected with Kaukolu, generating attention annotations (i.e., *Read* and *Skimmed* annotations). One such annotation can be seen in Figure 3.12.

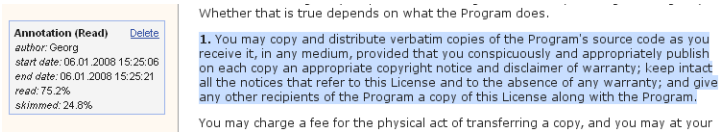


Figure 3.12.: Eyetracker annotation.

Tablet PC Use Case

For a Mymory use case involving a tablet PC, one rather exotic annotation type has been implemented, the *Drawing* annotation. It allows associating a small arbitrary drawing with text (Figure 3.13).

The context and PIMO integration has been shown already in Section 3.6.7.

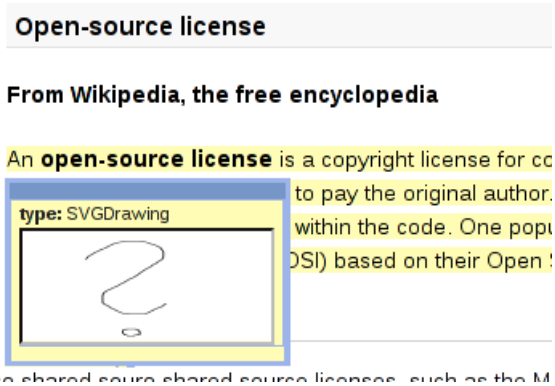


Figure 3.13.: Drawing annotations for the touchpad use case.

3.7.2. NEPOMUK

According to its Website[9]:

> NEPOMUK brings together researchers, industrial software developers, and representative industrial users, to develop a comprehensive solution for extending the personal desktop into a collaboration environment which supports both the personal information management and the sharing and exchange across social and organizational relations.

Gnowsis Prototype

Kaukolu was used as a supplementary component of the NEPOMUK Java-based Semantic Desktop prototype *Gnowsis* (Figure 3.14). For an explanation of the Semantic Desktop idea, see Section 2.3.

For each resource existing in Gnowsis, users were able to create a Wiki page using Kaukolu. The autocompletion feature of Kaukolu was connected with the Gnowsis RDF store, enabling users to quickly create links from the resource they were currently seeing to other resources.

In the context of NEPOMUK, several user studies have been carried out. The response to the Semantic Wiki was very positive in general. Here are quotations from the evaluations done in [Sau09] within the context of its Personal Semantic Wiki evaluation case study. There, users were asked to browse and relate resources of the Semantic Desktop, and use the Semantic Wiki to annotate resources. The Wiki could be used both as a tool for creating relations between resources as well used for entering comments for resources.

[9] `http://nepomuk.semanticdesktop.org/`

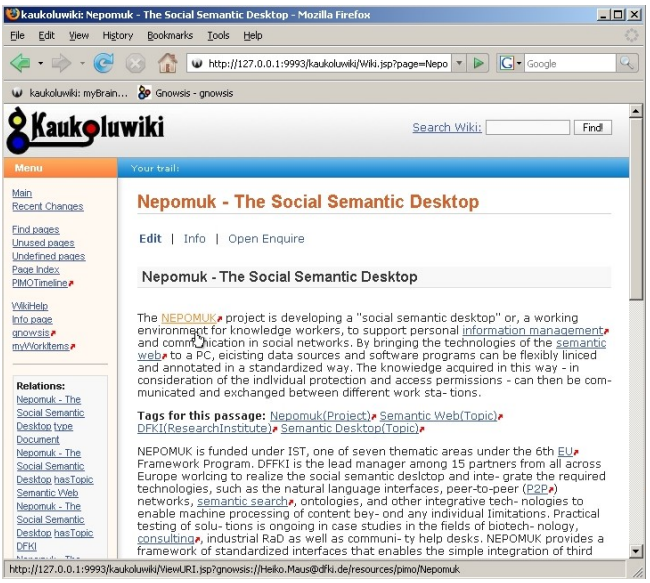


Figure 3.14.: SCETagTool results in the NEPOMUK Kaukolu Wiki component.

> (8.8.2, Conclusions PIM) *The majority of the participants used the Wiki as personal notepad.*
> [...]
> *The auto-completion feature helped to generate semantic links in a convenient and quick way.*
> [...]
> *As the Wiki was the second most used component and none of the users did not use it during the evaluation, it can be said that it fills the authoring gap and provides a different view of the users PIMO.*
> *The evaluation showed that the possible fields of application are manifold (documentation, comments on files, contact information to persons, to-do lists, notepad).*

Another case study in [Sau09] was concerned with evaluating long-term use of the Semantic Desktop for Personal Information Management. The Semantic Wiki was included in the Gnowsis Semantic Desktop prototype at some point. The following quote by a knowledge worker that had been using Gnowsis for Personal Information Management since years is relevant in that context.

> (9.3, Key Quotes) *"I seriously use gnowsis since it has the Wiki."*

3.7.3. RDF Views Example

In the context of a DFKI exhibition, an example for the way from documents and their annotations to views of the structured data contained in the annotations and back was built. It employs the DFKI OrgRep, the DFKI organizational repository, which contains information about employees and projects, using the PIMO. Several documents were imported and annotated with OrgRep concepts. Annotation of the documents was done using the ontology-based information extraction framework SCOOBIE [vEDAH09] (see also Section 2.9). Using templates for the RDF data, HTML views of the RDF data were built.

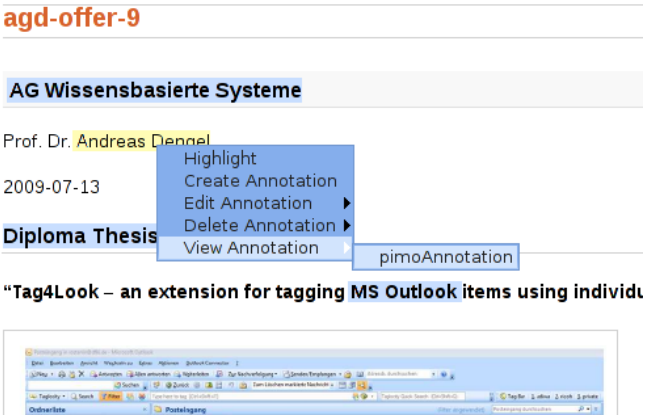


Figure 3.15.: Imported Web pages with added formal annotations.

In Figure 3.15, an annotated thesis offer is shown. There is an annotation of the occurrence of "Andreas Dengel", linking the text to a PIMO concept. The user is about to follow that link.

The user is then sent to the page seen in Figure 3.16. There, a template-based rendering of the RDF data of the PIMO concept is seen. The raw RDF data is also available on request, see Figure 3.17. The Wiki also shows the pages that reference the PIMO concept in question.

3.7.4. iGreen

According to its Website[10]:

> The iGreen project intends to develop and realize a network of location-based services and knowledge, which shall integrate various public and private information sources. This network shall allow

[10] `http://igreen-projekt.de/`

Figure 3.16.: Template-based rendering of RDF data contained in annotations.

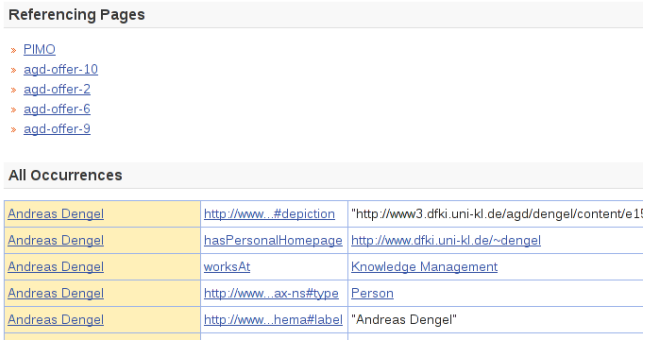


Figure 3.17.: Background RDF data.

> the realization of mobile decision assistant systems which facilitate the decentralized support and optimization of cooperative production processes, reaching improved energy efficiency, economic advantages, and environmental benefits.

iGreen is set in the agricultural domain. One challenge in iGreen was to make pest warnings (typically sent as PDFs or faxed in newsletter form) more accessible to farmers.

Automated Annotations

A software prototype was built that imports PDFs into Kaukolu and annotates the texts using concepts from agriculture ontologies (AgroVOC, AgroRDF, etc.). Geographical entities are detected and annotated using GeoNames. The result can be seen in Figure 3.18. The main iGreen component, the *OnlineBox*, issues a geo query to Kaukolu in case a field is shown; Kaukolu replies with a summary of the annotations found on any matching page, along with links to the specific paragraphs the annotations are located in (Figure 3.19).

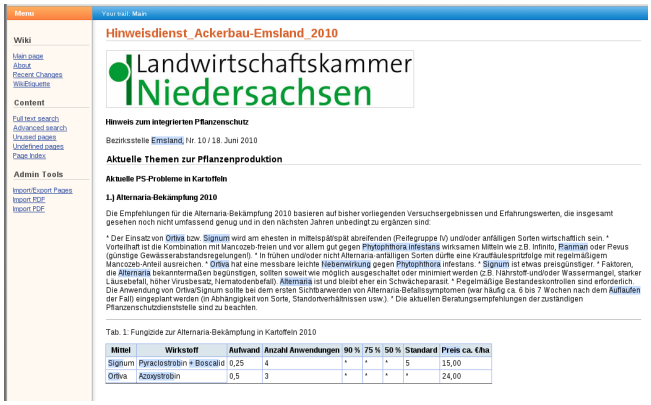


Figure 3.18.: An auto-annotated pest report in Kaukolu.

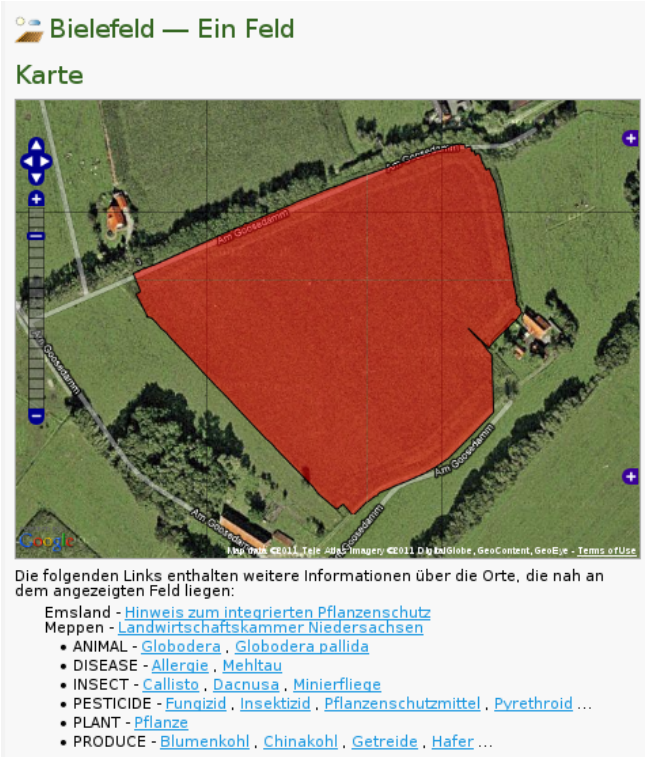


Figure 3.19.: Pest warnings showing in the central iGreen component, linking to Kaukolu.

In the end, this enables field owners to have a quick glimpse of what pest dangers are currently present for their fields.

3.7.5. Publications Management

In the following, Kaukolu's autocompletion feature and integration with other RDF-based applications is shown. This is also featured in [Kie06]. In this example, the user *Paul Miller* imports an ontology describing bibtex entries into Kaukolu, and adds a new publication item to a Wiki page using ontology–driven autocompletion. Then, he exports the bibtex RDF generated into an external application *RDFHomepage* [GSS06] which generates an HTML page containing a publication list.

Ontology Import

In Figure 3.20, a Wiki page holding the bibtex RDFS ontology used for the publication list is shown. In principle, Kaukolu can import any RDFS ontologies. On import, they will be converted to Kaukolu's RDF Wiki syntax. Note that one can now collaboratively edit the ontology within Kaukolu. Export to RDFS is also possible using the "View related RDF" button to the lower right. Updating the ontology can be done either directly in the Wiki or by re–importing the ontology.

Figure 3.20.: The Wiki page holding the bibtex RDFS ontology.

Adding Publication Entries

In Figure 3.21, the user adds a publication entry to his Wiki page. Since *hasType* (corresponding to *rdf:type*) implies that the triple's object will be of type rdfs:Class, only instances of rdfs:Class are displayed in the autocompletion suggestion box. The complete page describing Paul's publication item is shown in Figure 3.22. Note that one can use standard Wiki markup along with the RDF extensions (a bullet list is used).

Result

Exporting to External Applications

In Figure 3.23, an HTML page generated by *RDFHomepage* using the RDF formulated on the user's Wiki page is shown. Note that the page generated is intended for external audiences and cannot be edited. The RDF created in Kaukolu, a Java–based application, is passed to and processed by RDFHomepage, a PHP–based application.

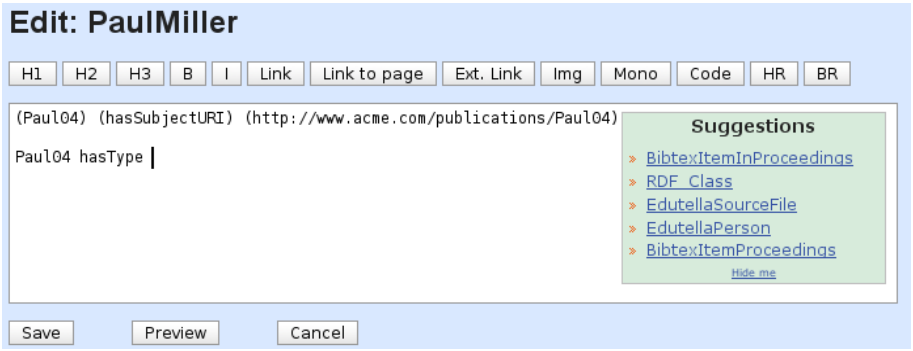


Figure 3.21.: Ontology–based autocompletion in action.

Figure 3.22.: The complete bibtex item.

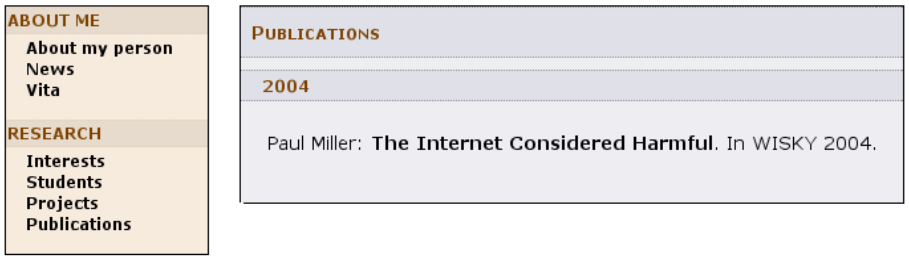


Figure 3.23.: The publications page as generated by RDFHomepage.

3.8. Conclusion

Multi-User Scenario

Annotations enhance the understanding and management of documents, especially when the documents are used by more than one person and for more than one context. Furthermore, annotated documents allow retrieval in use cases in which a keyword-based search is not enough, e.g., to find a document viewed a long time ago. Annotations allow additional filters to be applied to search, which enhances retrieval Precision—this holds particularly for contextual annotations and contextual filtering.

User Context

The annotation model of normal Semantic Wikis does not support multi-user annotations nor contextualized annotations. The Semantic Wiki prototype *Kaukolu* implements a novel annotation model that supports these types of annotations, and offers services building on this annotation model. Among these services is advanced search that lets the user search for document parts that have been read in a specific user context or document parts that have been annotated in a specific user context. Another new annotation-driven functionality is being able to switch to an annotation-driven structured view that allows novel ways of navigation among the documents available in the Wiki. Several practical use cases that make use of the novel functionality have been demonstrated. These use cases were part of several projects; Kaukolu was an integral part in the approaches of these projects, offering semantic extensions tailored for the project requirements.

Structured View

Machine-Interpretability

Most other Semantic Wikis focus on annotations that formalize the Wiki document content, making it machine-interpretable. This mostly allows to provide automated ways of creating structured overview lists. However, this is only a small part of what can be done with annotations. As shown in this chapter, capturing user context and user attention is key to advanced, non-intrusive improvements in navigation, search, and personalization. Capturing this additional user-specific information is not possible in standard Semantic Wikis due to their focus on formalizing Wiki content only. Kaukolu's extended annotation approach supports capturing and using this information, and more.

CHAPTER 4
Resource-Based Semantic Annotations

4.1. Motivation

Finding Items on the Web

Finding out what item (a book, a piece of computer hardware, a movie, a politician, a software library) suits your needs and tastes is a frequent task in everyday life and work. In the early days of the Web, completing such a task online was a challenge as the information needed was either not existing on the Web or impossible to find due to underpowered or nonexistent search engines.

The Challenge to Find High-Quality Information

Nowadays, many forums and other Websites with all kinds of domain-specific information exist. Users quickly find information sources typically with the help of search engines that will find many sources and hits, thanks to the vastness of the net. Thus, the main challenges have changed: They are (i) where to find *high quality* information, (ii) where to find *trusted* information sources, and (iii) where to find information voiced by people who have similar *tastes* and share the *opinions* of the user who is doing the search. Whereas the latter question is of little importance in technical domains (after all, finding a piece of computer hardware is mostly determined by hard technical objective requirements), for other use cases such as finding a book worth reading, the notion of *taste* becomes important.

4.1.1. Things and Their Properties

Building Blocks of the Web

The building blocks one encounters on the Web when collecting information about items one is interested in are *statements* about these *items*, voiced by different *people*. Yet, what are the building blocks search engines work with? Items, statements, or people, are concepts unknown to normal Web search engines—they work almost exclusively on *document* and *word* level[1]. Clearly, the notion of information sources and trust in these is missing. Compare this with an early vision of a global network, for example David Brin's *Earth*[2] — published in 1990, it

No Trust, No Individuals

[1] Some change is happening here, though, with specialized search engines for people and social network integration. However, this is proprietary and building on highly specific APIs typically.

[2] ISBN 0-553-07064-9 / `http://www.davidbrin.com/earth.html`

described a network very similar to the WWW, complete with forums and blogs, but also featuring "built-in" reputation and expression of trust. It is worth noting that in the Semantic Web Layer cake (Figure 2.1), trust is a key feature as well.

Ranking of Information

These days, the open internet is paying the price for not having reputation and trust ingrained in its underlying technology[3]: Spam is not only an issue in mails, but also on the Web. Ranking of information *should* be done by trust and relevance in the context at hand; instead, currently it is mostly done by popularity (cf. Google PageRank). Metadata that would allow moving from the level of "documents on Webservers" to "statements issued by individuals" is just not available—at least not in machine-readable form.

Shortcomings of Application-Level Solutions

One solution for this is to handle these issues on application level: Websites that have their own application-specific notion of individuals and reputation. In short, Facebook users and Facebook "Likes"; Amazon reviewers and item ratings. However, this leads to the somewhat sad state that interesting information is hidden in the backend database of large companies; users and use cases are restricted to the proprietary data model (and frontends) employed by these companies; while external services such as Web search can be provided, advanced services such as *information aggregation* over many different sources is not possible in practice.

4.1.2. A New Approach

In the following, an approach to introduce the notions of *items*, *people*, and *statements* to the Web get explained. This builds on Semantic Web principles, using ontologies to formalize these concepts. The approach uses facts that are available in machine-readable form, too, rather than the exclusively human-readable way the normal Web uses. *Semantic annotations* are the key to this.

Reputation and Trust

Several concepts such as *reputation* arise from the new data model. Formal reputation and trust metrics allow graceful handling of inconsistent information and lead to personalization of the user's view of the information in the system. Structured and semantic *comparison* of items gets possible. Several different *recommender* services intuitively fit into the approach. An important part of the approach are *annotation recommenders*—recommenders that help the user with creating annotations. These recommenders are intrinsically *goal-directed* in order to optimize the data available efficiently for the recommendation services the user is interested in—item recommendation typically.

[3] Note that here, by *trust* the belief of the user in opinions voiced by others is meant, not trust from a security perspective, which is handled by HTTPS etc.

Explanations

The high formality of the system allows comprehensive explanations during recommendation tasks and statistics, while at the same time the ability of the system to cope with subjectivity and disagreeing statements makes it resilient both against annotation errors and spam.

In the following, a summary of the system ideas is shown (Section 4.2). Then, an in-depth explanation of its building blocks and features is given (Section 4.3). A research prototype implementing these ideas, *Skipforward*, is presented in Section 4.4.

4.2. Core Ideas

The basis of the approach is to design and employ a data model tailored for the task at hand—finding a balance between a model that sacrifices a lot of semantics to other requirements (such as the pure WWW/HTML approach or standard Wiki approach would do), and a data model that is too strict to be user-friendly (i.e., very formalized or brittle). In fact, a data model that allows the user to express uncertainty would be beneficial.

Key Concepts

After considering many options, analyzing similar systems, and contacting other researchers, a simple but versatile data model was conceived that is based on the following concepts:[4]

- *Item Types* that form a class hierarchy, enabling discerning between books, people, etc. (plus their *instances*, the actual books, etc.)
- *Feature Types* that also form a class hierarchy, describing the set of possible features (e.g., "Expensive") that items can be associated with (plus *instances*, the actual user opinions). A keystone is that any *feature*, among other metadata, is given a truth or *applicability* value, representing one approach of implementing negation.
- *User* – Each (instance of) an item type/feature type belongs to a user.

To illustrate this model, here is a small example:

- There is an item type Book, created by user Arnold.
- There is an item Le Petit Prince, an instance of the Book item type. The item belongs to Arnold as well.
- There is a feature type Interesting, created by Arnold.

[4]The model is explained in more detail in Section 4.3.1; this is intended as a short overview.

- *Arnold* also supplied one opinion: He created an instance Interesting_1 of the corresponding feature type[5]. This instance carries applicability +1.0.

The end result of this set of data is this:
"Arnold thinks that Le Petit Prince is an interesting book."

Now, add more information:

- *Berta* creates an instance Interesting_2 of the corresponding feature type. This instance carries applicability **-1.0**.

This additional data says: "Berta thinks that Le Petit Prince is *not* interesting."

Opinion Clash

This opinion is at odds with Arnold's opinion before; a *clash of opinions* has formed. What is a user interface expected to show in this case? It might present all individual statements (not scalable to many users and statements), or aggregate all individual statements for each feature type, averaging the applicability ratings. But clearly, this should be done using a *weighted* average—with these weights representing *trust* in each of the contributing people.

Weighting Opinions

[GHS08] defines trust as a multi-relational concept:

> "A *truster* trusts a *trustee* (e.g., a person, an institution or a technical system) in a certain *context*, if the truster has confidence[6] in the *competence* and *intention* of the trustee and therefore beliefs that the trustee acts and behaves in an expected way, which does not harm the truster."

Then, Gutscher distinguishes trust into two categories:

Competence trust Trust in the *capability* of a person, in an institution or in the functionality of a machine or a system.

Intentional trust Trust in the *moral integrity* (benevolence) of a person.

In the following, *trust* and *competence* are used somewhat synonymously—one could also say that we are talking about *trust in the competence* of others. *Intentional trust* (or the lack thereof) is not considered; however, malevolent behavior, in the outlined system, just looks like a user with strange opinions.

One question now is how to derive trust values. A possible approach is to compare statements of users and suppose that in case two users agreed a lot

[5] Note that for Feature Instances their labels are of no consequence.

[6] Although the terms trust and confidence are often used synonymous, they are not exactly the same [Ada05].

(they have a high *user similarity*), they trust each other. Another approach is to rely on explicitly stated trust: for example, a user can state that he trusts another user (the *Web of Trust* models of encryption software comes to mind).

A welcome side effect of using trust as weights in the aggregation formula for individual opinions is that each user has a completely **personalized view** of the data in the system. This is explained in more detail in Section 4.3.3. This personalized view also allows improved recommendations (H3).

This concludes the short overview of key ideas. In the following, the key components of the approach get presented in more detail.

4.3. Designing a Resource-Centric Annotation System

4.3.1. The Top Level Ontology—Skipinions

There is a number of requirements the top level ontology (coined *Skipinions*) shall be able to handle (see also [KM11]). These requirements have been derived by a study of existing non-semantic systems (e.g., the *Brettspielbrowser* mentioned in Section 4.5.2, TVTropes.org, etc.) and discussions with domain experts. User feedback (see Chapter 6 and Appendix D) confirmed these decisions as well.

1. *The ontology should be able to represent user opinions of items (e.g., books, movies).*
   This is solved by providing an Item and a Feature class: for example, book features could be "Thriller (Genre)" or "Fast-paced writing style". Every Item can be associated to a Feature by using the Item's *hasFeature* property.

2. *As much information as possible should be machine-readable.*
   Any Feature or Item instances are instances of the Feature Type or Item Type classes which are organized hierarchically and, thus, carry semantics. Feature types that are used purely for structuring or grouping purposes can be marked as abstract.

3. *Provenance of statements needs to be tracked.*
   This can be solved by assigning an individual namespace to every user. Then, the URI of every instance created by this user has to use this namespace. This also fits nicely with Linked Open Data principles (see Section 2.2).

4. *The facts databases of multiple users should be easy to merge.*
   Fact databases can be just copied together using this approach. In case several users have created different RDF resources for semantically identical items, the *owl:sameAs* predicate can be used to mark these resources as equal.

5. *Plain text comments should be supported.*
   Internationalized plain text comments can be added to any Feature.

6. *It should be possible to explicitly dissent with an opinion of another user.*
   This is implemented by the *applicability* property of every Feature. Applicability -1 means "this feature type does not apply to this item", applicability $+1$ means "this feature type applies to this item", implementing the Open World Assumption. Applicability 0 means "it is not possible to say whether this feature type applies to this item". Additionally, any Feature can point to another Feature instance, implementing discussion threading.

7. *Marking an opinion as uncertain should be possible.*
   This is implemented by the *confidence* property on every Feature. Together with *applicability*, this implements an approach similar to the Dempster-Shafer evidence theory.

8. *Statements referring other items or literals must be supported.*
   For example, this is needed to link books to their authors. It can be implemented by adding arbitrary RDF statements to each Feature instance. In the following, these feature types are called **non-binary feature types** in contrast to normal "binary" feature types (whose instances state that a feature applies or not, depending on its applicability value).

9. *The amount of noise seen by users should be kept minimal.*
   The feature hierarchy as presented by the system is created by the owner of the respective namespace. Arbitrary changes of the feature hierarchy are discouraged. This is both a limitation and a feature of the system. It is limiting insofar as users cannot create new feature types on the fly. On the other hand, systems that implement this (i.e., most normal tagging systems) show that a *lot* of entropy enters the system otherwise. In this approach, the noise gets limited to the feature instance level where it can be handled in a coherent manner.

An overview of the Skipinions ontology structure can be seen in Figure 4.1. Note that the concept of the *user* is not shown there explicitly. Provenance of state-

ments can be derived from the namespaces of the RDF subjects of any fact in the system: Any RDF triple that is part of any fact created by the user Alice will have a subject URI that belongs to Alice's individual namespace.

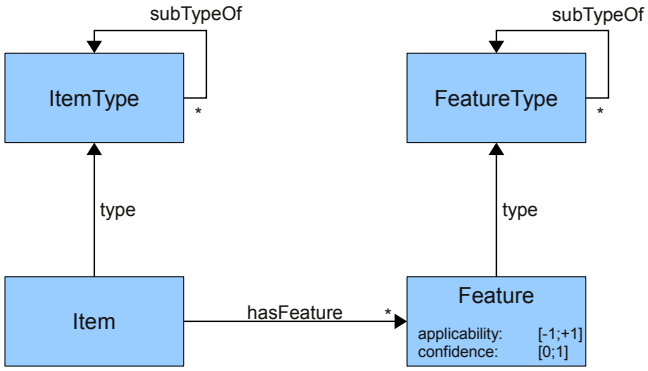


Figure 4.1.: Class diagram of items and features.

Formal Definitions

In later sections, the following definitions represent the basis for the formulas presented.

Every **feature** f is a tuple (a, c):

- a represents the applicability or degree of truth
- c represents the confidence in the feature statement

For each user/item/feature type combination, the approach can consider at maximum one feature[7]. The function f_b (for binary feature types) or f_{nb} (for non-binary feature types) keeps track of this assignment.

$$f_b : U \times I \times T_f \longrightarrow [-1; +1] \times [0; +1] \cup \{\varnothing\}$$

$$f_{nb} : U \times I \times T_f \times O_{t_f} \longrightarrow [-1; +1] \times [0; +1] \cup \{\varnothing\}$$

- U is the set of all users
- I is the set of all items

[7]Nothing keeps the user from creating several instances of one feature type for one item. However, in practice it is prudent to treat this as *opinion updates* then, and only including the newer feature in calculations.

- T_f is the set of all feature types
- O_{t_f} is the set of allowed objects/literals for feature type t_f
- $\varnothing$ is *undefined*

This can also be written as a variable with indices:

$$f(u, i, t_f) \overset{equiv}{=} f_{u,i,t_f}$$
$$f(u, i, t_f, o) \overset{equiv}{=} f_{u,i,t_f,o}$$
$$\text{with } i \in I, u \in U, t_f \in T_f, \text{ and } o \in O_{t_f}$$

4.3.2. Domain Ontologies

The top level ontology lays the foundation for formalizing opinions and associating these with items. However, it contains only abstract classes. For expressing actual opinions, both information about what item types may occur and what feature types may occur is needed. This knowledge is domain dependent, and as such, must be contained in domain ontologies that typically subclass classes of the top level ontology. Describing such domain ontologies is out of scope in this chapter. However, in Section 4.4, several examples of domain ontologies are shown.

4.3.3. Building Personalized Views: Trust and Confidence

No Superuser

As outlined in Section 4.3.1, the proposed system does not feature a "ground truth" concerning features of the items; there is no "superuser". Rather, it carries lots of different statements or *opinions* of different users with different tastes and different background knowledge concerning these items. Thus, there exists no "one true view" of the information in the system.

Trust

Following the notions presented in Section 4.2, the key is to embrace this fact, as it is, in fact, a strength of the system. While other approaches prevent clashes in opinion or just aggregate statistics (see up- and downvotes in Social Media), the rich data model of this approach allows a more informed way of aggregating opinions (see Table 4.1). By *aggregating opinions*, the act of gathering many individual statements and presenting them as one statement is meant. In the context of this approach, this means considering one item and one feature type, and gathering all features available for this, by any user, then calculating *one* "aggregate opinion" that is supposed to match the current user's opinion as

	CF	CBF	This approach
Number	1	many	many
Types	rating	content	anything
Subjective	yes	no	yes

Table 4.1.: Differences between features in collaborative filtering (CF), content-based filtering (CBF), and this approach [Mit08].

closely as possible. For aggregating individual opinions, weights have to be used; this weight represents *trust* in one user (with regard to one feature type). The result is a completely personalized view of the data in the system [ZL04]. There are several ways to determine trust.

User Similarity

In collaborative filtering (see Section 2.6), trust[8] is derived from *user similarity* with regard to item liking [MA04]. Adjusted to the data model of this approach, this can be read like this: Trust[9] can be derived from user similarity regarding that certain feature type. Or, to give an example: Arnold knows that Berta's opinions concerning humor for books both read are similar. Therefore Arnold *trusts* Berta's judgment concerning humor, i.e., Arnold will trust Berta in case she says a book she has read but he did not yet has a lot of humor in it.

Specific Trust

Trust Maths

Trust, in mathematical terms, may be discrete ("not trusted", "fully trusted") or continuous [Mau96, Mar95]. Discrete values are most suitable for display and manually assigned trust. Continuous values can be transformed to discrete values by rounding. This is useful for displaying purposes after calculations, in case perfect accuracy is not of importance. As [GHS08] mentions, continuous values can be hard to be interpreted as in contrast to discrete (and, possibly, paraphrased) values, they carry no explicit semantics. Some semantics for continuous values are unclear: Are two statements with trust value 0.5 "worth" as much as one statement with trust value 1.0?

Confidence

Also, there is another element relevant in the context of trust: In case trust is derived, not stated explicitly by the user, the amount of data that formed the basis of the derived trust value becomes important. To come back to the example above: If Arnold and Berta agreed concerning their opinion regarding humor for exactly one book only in the past, the user similarity is high; still, the actual trust between them will probably not be. This hints at another element hidden in trust values, *confidence*, which gets addressed shortly.

[8] Here, *trust* means that another user has a similar taste, and, thus, is a good source for item recommendations.

[9] Here, *trust* means that another user has a similar taste with regard to a certain feature type, and, thus, is a good source for information with regard to that feature type.

In the following, a way to determine user similarity and derived trust in the context of the proposed data model is described. It will be discussed how to determine and handle confidence in the derived trust values. Then, the actual aggregation process is described. Details about these topics can also be found in [Mit08].

User Similarity and Deriving Trust

Pearson Correlation

The *Pearson product-moment correlation coefficient* is a measure of the linear correlation between two variables. In Collaborative Filtering it is often used for determining user similarity with regard to their item ratings (see Section 2.6.1). The standard Pearson correlation, if applied to this approach, has some drawbacks: for example, it suffers from sparsity problems, as it needs more than one statement per feature type in common to calculate the correlation value. This is because the standard Pearson correlation normalizes the scale of the input variables. In this approach, this is of little benefit—in contrast, normalization might even be bad since applicability values do not use a scale such as 0,...,10, lacking any semantics, but rather -1 (does not apply) to $+1$ (applies completely), with 0 being the neutral element: one cannot say whether this feature type applies to the item or not[10]. The normalization step typically takes care of users using only part of the scale (e.g., assigning ratings from 4 to 9 only); however, in this approach this would disturb the semantics, as the neutral value might shift. Also, it would be beneficial to have a correlation measure that works with only one common statement of the two users in focus.

Normalization

Constrained Variant

[SM95] introduces the *Constrained Pearson Correlation*. This variant leaves away the normalization step and produces useful results with one common rating already.

When applied to this approach, Formula 4.1 results. This formula includes weights that allow handling confidence values.

I_{xy} denotes the set of items annotated both by user x and user y. $r_{x,i}$ denotes the rating (in the context of this approach, the applicability) user x has assigned to item i. $w_{x,i}$ is the weight (in the context of this approach, the confidence) user x has expressed in his opinion concerning item i. $w_{xy,i}$ can be derived from $w_{x,i}$ and $w_{y,i}$ in various ways. Finding a good function for $w_{xy,i}$ is dependent on the use case. The minimum and geometrical mean functions have been evaluated (see Chapter 6).

[10] Note that applicability 0 does *not* mean "I am not sure"; this is what the separate confidence value in features is for.

$$sim(u_x, u_y) = \frac{\sum_{i \in I_{xy}} w_{xy,i} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_{xy}} w_{x,i} (r_{x,i})^2 \sum_{i \in I_{xy}} w_{y,i} (r_{y,i})^2}} \tag{4.1}$$

It has to be noted that in this approach, all calculations are done per feature type. Thus, the result of this function is the user similarity of users u_x and u_y with regard to a specific feature type.

Incremental Calculation

In [Mit08], an implementation of this approach that features an incremental algorithm is described. The incremental algorithm prevents having to re-calculate similarities for the whole database on each change, instead only recalculating user similarities influenced by the facts that have changed.

Explicit Trust

In general, other sources for deriving trust are possible; a simple notion would be explicit trust, i.e., letting the user manually assign trust values to the other users (or, to be exact, to every combination of user and feature type) of the system. These different approaches can be combined: for example, explicit trust (if stated) can override trust values derived from user similarity. Many existing systems use explicit trust exclusively: Think of the *Web of Trust* approaches common for encryption software, or subscription/like mechanisms in social networks, which in some ways implement similar behavior. However, setting up explicit trust is tedious, and often users have not enough information to actually answer questions about explicit trust the system might require. Two strong advantages of derived trust are that (i) derived trust has explicit grounding, and (ii) derived trust can detect and reflect trust changes happening over time. In the latter case, explicit trust values are potentially hazardous: Even if there is obvious evidence that trust is not warranted anymore, explicit trust gives the (now potentially malicious) person high weight. Combinations of approaches are possible in principle: for example, notifying a user that the explicit trust assigned to a user results in bad applicability predictions is possible, as is allowing the user to give *relevance feedback* for specific instances of opinion clashes.

Advantages of Derived Trust

Confidence in Trust Values

When aggregating opinions, these concepts play a major role:

- **Trust/Competence** – can I trust the statements by a person?
- **Confidence** – how confident am I that I can judge the competence of a person?

Trust/ Competence

Trust/competence is derived from user similarity, which can be calculated using Formula 4.1. *Confidence* in the trust metric, however, is not addressed by the Constrained Pearson Correlation. In this context, confidence is mostly dependent on the amount of data that was available for running the Constrained Pearson Correlation: If we have only two statements by two users (with regard to one feature type) that are the same, confidence in the correlation derived is intuitively much lower then if, say, 1000 exactly matching statements were present.

Measuring Confidence

A possible measure for confidence in trust values is shown in Formula 4.2. Ignoring weights/confidence values, it calculates two times the number of items annotated by both users divided by the number of items annotated by user one plus the number of items annotated by user two. Complying with the requirements for a confidence function, its range is $[0..1]$. Other functions have been considered in [Mit08]; this function's characteristics for a number of cases make it well-suited for the task. Variables are the same as in Formula 4.1.

$$conf_{sim}(u_x, u_y) = \frac{2 \sum\limits_{i \in I_{xy}} w_{xy,i}}{\sum\limits_{i \in I_x} w_{x,i} + \sum\limits_{i \in I_y} w_{y,i}} \tag{4.2}$$

Feature Aggregation

The calculated trust and confidence values allow aggregating individual statements with regard to one feature type to get aggregated into one personalized weighted statement. This can be done using the following formulae. Note that all formulae's inputs are specific to the currently selected feature type; the reference to t_f is left away for readability reasons.

u_x denotes user x, $conf(u_x, u_y)$ has been explained in Formula 4.2, and $c_{x,i}$ and $a_{y,i}$ are confidence and applicability of users x and y in their statements concerning item i (and the implicit feature type t_f).

$$aggr_{app}(u_x, i) = \frac{\sum\limits_{u_y \in \hat{U}} comp(u_x, u_y) conf(u_x, u_y) c_{y,i} a_{y,i}}{\sum\limits_{u_y \in \hat{U}} comp(u_x, u_y) conf(u_x, u_y) c_{y,i}} \tag{4.3}$$

$$aggr_{conf}(u_x, i) = \frac{\sum\limits_{u_y \in \hat{U}} comp(u_x, u_y) conf(u_x, u_y) c_{y,i}}{\sum\limits_{u_y \in \hat{U}} comp(u_x, u_y) conf(u_x, u_y)} \tag{4.4}$$

with

$$\hat{U} = \{u \in U | f(u,i) \neq \varnothing\} \quad \text{and}$$
$$f(u_y, i) = (a_{y,i}, c_{y,i})$$

and

$$comp(u_x, u_y) = \frac{sim(u_x, u_y) + 1}{2}$$

Some remarks concerning these formulae:

- $comp(u_x, u_y)$ is the *similarity projection function* necessary for transforming the $[-1.. + 1]$ range of $sim(u_x, u_y)$ to $[0.. + 1]$. In the evaluation (see Chapter 6), several functions are considered.
- $\hat{U}$ is the set of all users that have assigned a feature of the feature type in question to the current item.
- $aggr_{app}$ is the resulting aggregated applicability.
- $aggr_{conf}$ is the resulting aggregated confidence.

In the end, both formulae represent a weighted mean including trust into the weight.

4.3.4. Inferencing

Data Sparsity

For most functionality and algorithms of interest in the domain, it is desirable to have as much data as possible in the system. However, as in normal circumstances most data will be user-generated, it is to be expected that data sparsity is a problem. *Data sparsity* means the fact that most users will annotate only a part of the items known to the system, and even more important, that for these items, only a small part of the known feature types will be annotated.

Subclass Hierarchies

However, this problem can be remedied to some degree. The system is aware of some semantics of the feature types and, thus, their instances: The subclass hierarchy present for the feature types.

Inference

For example, assume there are two feature types *Science Fiction Genre* and *Hard Science Fiction Genre* (that can be associated to stories), and the latter one is a subclass of the first. This is illustrated in Figure 4.2. Then, an instance of *Hard*

Science Fiction Genre implies the statement that the same item belongs to the *Science Fiction Genre* as well—if the instance has positive applicability. It has to be noted that this is not true if the applicability is negative: If the fact exists that the story in question is *not* hard scifi, the same story can still be normal scifi. However, if the fact exists that a story is *not* normal scifi, the fact that the story is *not* hard scifi *does* follow.

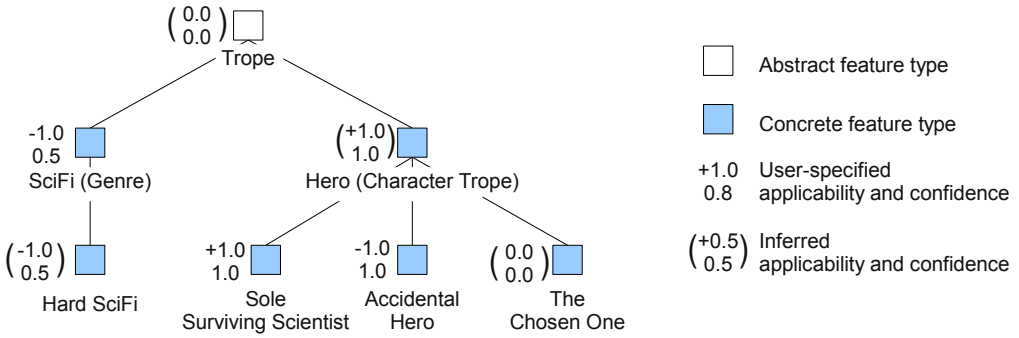


Figure 4.2.: Inferencing.

Inferencing and Applicability

In general, this means that features with positive applicability allow inferencing *up* the subclass-of hierarchy, whereas features with negative applicability get inferred *downwards* in the subclass-of hierarchy. For abstract feature types (see Section 4.3.1), no instances are inferred, and inference traversal gets stopped. There, downwards inference of negative applicability can be seen for the genre part of the hierarchy; upwards inference of positive applicability is visible for the character trope part. *Trope* is an abstract feature type and, thus, is ignored by inference; since *The Chosen One* has no user-assigned instance and its superclass *Hero* has inferred positive applicability (which does not get inferred downwards), applicability and confidence zero are assigned by inference. This represents the statement "The system does not know whether a *Chosen One* occurs", which is intuitively correct; from the fact that a *Hero* occurs, it does *not* follow necessarily that that *Hero* is a *Chosen One*.

In case several feature instances exist for subclasses of a feature type that has no instance, the highest applicability and confidence is used for inference of the new feature instance. With additional knowledge about the feature types, other types of inferencing are possible as well: for example, there can be explicit *opposite-of* relations between feature types.

4.3.5. Item Recommender

Goals

A key functionality of resource-based systems is *item recommendation*, i.e., listing items that are potentially interesting to the user. Of course, the relevance of any item depends of the recommendation context: Should the recommendation be about what book the user should read next for entertainment? Or should the recommendation be about what book to read for learning more about a specific topic? Should books by the same author be recommended, for learning more about the author's writing style? Or, quite the opposite, should books on similar topics be recommended *written by other authors*, to broaden the reader's horizon?

In these scenarios, recommenders based on collaborative filtering struggle, as the sole data they base their recommendation on is the liking/rating of users. Consequently, these recommenders can only recommend works that users with presumably similar tastes have liked, too.

Tuning Recommendations

The item recommender in this approach can be tuned to search for any combination of feature types. Since *liking* can be included as a feature type, the recommender is essentially a superset of the typical collaborative filtering recommender. For each recommendation, specific explanations can be supplied as well. This allows the user to fine tune recommendations: If the recommendations for some reason do not satisfy the user, he can look at the explanations, and adjust the recommendation criteria accordingly.

A concrete implementation of an item recommender based on this approach is shown in Section 4.4.2.

The item recommender works best if items (i) are annotated with instances of many different feature types, and (ii) many items share a set of feature types they are annotated with. These two requirements become relevant for annotation recommenders later.

4.3.6. Expert Recommender

Recommending *experts* (i.e., people that have a high trust value with regard to a specific feature type) can be done by ordering users by trust values. This becomes most interesting to users for feature types that have subjective nature: "Show me other users of the system who agree with me regarding whether *humor* is present in stories or not.".

The expert recommender works best if there is a high *annotation overlap* between users, i.e., users have created instances of the same feature type for the same items. Otherwise, user similarity, and thus trust values, cannot be derived.

4.3.7. Annotation Recommenders

Challenges When Annotating

Creating annotations is typically a tedious process. Not only is finding the correct annotations (in this case, finding the correct feature types to express the notion the user has in mind) time-consuming; also, users are often hesitant to create a new annotation as "it might not be correct". This approach tries to alleviate the latter problem by allowing the user to set a low confidence. Still, user support while creating annotations is possible, in a form that allows quick creation of annotations, and giving the user some kind of visible feedback, or *instant gratification*.

Goal-Directed Recommenders

Annotations are not an end in itself. Other functionality such as item and expert recommenders rely on them. Therefore, the task is not really to get users to annotate just "a lot"; instead, users should get guided to create annotations that help existing system functionality. This also plays well with the idea of instant gratification: If the user creates exactly the annotations that "help" the item or expert recommender, results (i.e., improvements in the recommender output) will be visible quickly.

This leads to the fact that the *annotation recommenders* needed are intrinsically *goal-directed*: Depending on what functionality is targeted for improvement by the user creating annotations, different sets of feature types are presented to the user as candidates for types for new feature instances.

In the following, several annotation recommender approaches get described. Apart from the recommender basing its recommendations on similar items, these have been implemented and tested in detail in [Ami12].

Annotation Recommender: Item

This annotation recommender functionality aims at improving item annotation with the goal of optimizing results of the item recommender (Section 4.3.5). It taps several other information sources, and bases its recommendations potentially on several criteria.

- **Features present for similar items** – feature types that are present for items returned by the similar item recommender but that are not present for the current item are suggested. In case the presumably similar item is not actually similar, this will help discerning it from the current item; in case it is really similar, this approach will strengthen this relationship.
- **Feature types known to the user** – users will focus on a set of individual "most interesting" feature types when creating annotations. Building on this set of types and not confusing the user by recommending feature types

probably not known to him is an improvement. Evidences for the fact that a feature type is known by the users are (i) the user has created instances of the feature type in question before, and (ii) the user has read a description of the feature type (e.g., visited the description page of the feature type).

- **Feature types used often by all users** – this evidence source helps making the items in the system comparable.
- **Web search** – the idea for this evidence source is to do a Web search with a query fitting the item in question, then extracting evidence for occurrences of a feature type in the resulting documents. This approach poses several challenges: (i) the Web search query has to actually match the item. Typically, this is domain dependent; for example, for books one might search for the book title plus the author's name. (ii) parsing the HTML documents found is difficult: for example, in the book/story domain, Web pages often list related items and authors in their navigation bars; these parts of Web pages should be ignored in the feature type evidence extraction step that follows. (iii) extracting evidence for feature types is non-trivial. Without external knowledge, the only "semantic bridge" from the RDFS class representing a feature type to the string representing the documents found is the label of the RDFS class.
- **Feature type co-occurrences** – if there exists a reasonably sized (external) corpus of items and associated features, it is possible to calculate feature type co-occurrences (i.e., information about what feature types often occur in tandem with what other feature types), and basing recommendations on this information.

Section 4.4.2 explains a concrete implementation of this recommender and its evidence sources.

Annotation Recommender: Expert

This annotation recommender functionality aims at improving expert recommendation (Section 4.3.6) quickly. As mentioned, computing user similarity for deriving trust values is mostly dependent on annotation overlap. This annotation recommender proposes feature type/item combinations to the user that increase annotation overlap. Several requirements exist that can be used as hints for recommending feature types:

Requirements For Candidates

1. There must not be a feature instance for the feature type/item combination of the recommendation already.

2. The item of the recommendation must be known to the user (that is, the user previously annotated the item).

3. The feature type recommended must be known to the user (that is, the user previously created instances of the feature type).

Ranking Candidates

The remaining recommendation candidates can be ranked by the following criteria:

1. Feature types that have been previously annotated by users that are missing correlations with the current user for that feature type are most important.

2. For ties, feature types with no existing correlations between the current user and all the other users annotating items with that feature type should get preferred.

3. For remaining ties, feature type/item combinations that maximize the number of additional correlations between the current user and other users should get preferred.

Again, see Section 4.4.2 for a concrete implementation.

4.4. Prototype Implementation: Skipforward

In this section, *Skipforward*, an implementation of the ideas of this chapter, is presented. In the following, it is described how each functionality and approach presented before has been put in practice. Furthermore, Skipforward implements some features relevant for usability and accessibility that are not of relevance for the general approach (e.g., RSS feeds, and a Linked Data Interface)—these features get explained as well.

4.4.1. Design Choices

In the broad approach outlined in the sections before, some interfacing components and implementation choices were missing. Some of these aspects are covered in the following.

Code Design

Few, Simple Frameworks

The Skipforward prototype was implemented from scratch, with a focus on using just a few simple frameworks, and keeping most of its internal design easy to understand. This was necessary as it was intended as a long-running project with, over time, different people working on it. To minimize the time required to get to know the system, functionality was implemented mostly in plain imperative Java. Declarative approaches were avoided; while often elegant and concise, their costs are higher familiarization time, and more complex debugging and tracing. In the end, the key points a programmer has to know when starting to work on Skipforward take up about one page of text, which can be found on the Skipforward Web pages[11].

Web Application

Apache Jena

Skipforward is implemented as a Web-based application. It uses Apache Jena TDB[12] as its RDF triple store. While TDB is still work in progress, it was known beforehand to scale reasonably well, and have a clean interface. During implementation, steps have been taken to minimize any problems caused by instabilities: for example, TDB occasionally corrupts its on-disk binary database on out of memory problems or JVM crashes; to make this less problematic, an N-TRIPLES backup file of each user's data is kept and regularly updated which can be automatically recovered from. This backup file also proved very helpful for many non-recovery purposes such as statistics and easy debugging/patching of user data. On the demonstration server, backup files are auto-committed to Subversion[13] which allows easy monitoring and history keeping.

Java RDF Beans

For making access to the Skipinions data structures easier, a custom simple Java bean approach was implemented. This is based on four Java classes corresponding to the ItemType, Item, FeatureType, and Feature Skipinions classes. These Java classes provide a lot of convenience methods.

StringTemplate

For the user interface, a model view controller approach was implemented. Views are built in custom Java code based on custom Java beans; rendering to HTML and RSS is done using the StringTemplate library[14]. StringTemplate is very small and has, by design, limited functionality: for example, it does not allow code in its templates and, thus, enforces proper MVC patterns. This was a lesson learned from Kaukolu/JSPWiki that uses Java Server Pages (JSP) for templating instead, with many of its templates containing nontrivial Java code, severely breaking the MVC pattern and causing unforeseen problems.

[11] http://skipforward.opendfki.de/wiki/DocumentationForDevelopers
[12] http://jena.apache.org/
[13] http://subversion.tigris.org/
[14] http://www.stringtemplate.org/

jQuery

The browser frontend side, finally, uses the jQuery JavaScript library[15] for interactive components where necessary. Communication with the backend is mostly restricted to loading HTML; only for some functionality such as auto-completion, JSON is used.

An early version of Skipforward used the Dojo toolkit[16] as JavaScript library; however, that version did not support using multiple browser tabs, and Dojo, at that time, was evolving quickly, making updating processes very tedious.

System Design

Peer to Peer

Concerning storage of the data, a system based on Skipinions allows several options. One option is to build a centralized system with one database, and allow the users connect to some central place. Another option is to implement Linked Data principles (see Section 2.2), i.e., connecting to other sites using HTTP, and crawling RDF data from remote sites, which would implement a kind of *Skipforward cloud*. However, crawling is slow, and this approach has limited options for two-way communication between peers. Instead, a custom XMPP[17]-based approach was chosen. XMPP is a routing protocol for XML, normally used for online chat and presence exchange. It allows realtime exchange of arbitrary XML messages and other data. This makes it well suited for data exchange and synchronization of distributed data nodes. In Skipforward, every used can potentially run his own Skipforward node that connects to other nodes via XMPP, with the XMPP contacts list (or "roster") representing the list of known peers. The data transfer model between nodes is full replication.

User Interface

The user interface is Web-based. This keeps the system operating system independent and flexible with regard to users connecting from different devices: for example, it was easy to adapt the HTML-based user interface for usage with smartphones and tablets.

Generality

While it was desirable to have a usable and clean interface, it was necessary to be able to cover multiple application domains (e.g., books, board games, music). This called for a user interface that is generic enough to support these different domains, yet does not sacrifice usability. The Skipinions ontology already represents a reasonable balance between abstraction and user-friendly concepts; this approach was pursued in the general user interface as well.

RSS Feeds

For letting the user keep up to date, Skipforward supports RSS feeds not only for recent changes, but also for complex search. This means that for any search

[15] `http://jquery.com/`
[16] `http://dojotoolkit.org/`
[17] `http://xmpp.org/`

operation, an RSS feed can get generated, informing the user of any changes in these search results over time.

Multilinguality

From the start, it was clear that the system will be used by people with different native languages. Since multi-language capability is built-in in RDF, supporting different languages is not a problem in general. Skipforward supports selection of the primary language and secondary language(s) when logging in. For any information present in the system (feature types, item types, comments, etc.), preferably the primary language is shown, or, if not available, one of the secondary languages.

4.4.2. System Outline

In the following, the building blocks of the system are described: its ontologies, the user interface, and recommender functionality. Since 2008, a Skipforward demonstration installation has been running at `http://skipforward.net/`. While most of the details explained here are valid for the Skipforward implementation in general, some details apply to the skipforward.net demonstration installation only. For example, availability of ontologies depends on what installation gets used, and what nodes the Skipforward node is connected to. References to the source code and resource files in this section are valid for revision 963 of the code.[18]

Ontologies

The Skipinions ontology only defines basic concepts and cannot be used by itself for any real items. Instead, domain ontologies extending Skipinion's concepts are used. These subclass the Item and Feature classes of the Skipinions base ontology.[19]

Currently, skipforward.net covers multiple domains concerning the items it can host opinions about. Apart from ontologies that model features of board games and a corresponding set of instances, *DBTropes* is used mainly. A detailed explanation of DBTropes is available in Section 5.7; here, it suffices to say that DBTropes.org is a wrapper of TVTropes.org, a Wiki describing works of fiction by associating features—known as "Tropes"—to these works. The focus of TV Tropes is providing content-based annotations (as opposed to more technical information as, for example, supplied by IMDb.com), with a definite emphasis

[18] `http://skipforward.opendfki.de/browser/trunk/skipforward?rev=963` / October 2013

[19] Skipforward's base ontologies can be found in its source directory `/ontology/`.

on fun and entertainment aspects. DBTropes extracts the information contained in the TV Tropes Wiki, and publishes it as Linked Data. The implicit data model used in the TV Tropes Wiki matches the data model used in Skipforward quite well. DBTropes uses the Skipinions ontology for its output, and Skipforward can potentially consume this data directly. This data is mainly used as a source for feature types and the hierarchy within feature types.

DBTropes Conversion Bot

Instead of directly connecting with DBTropes, there is a **DBTropes conversion bot** that provides an interface between skipforward.net and dbtropes.org. This is necessary since DBTropes is very large, and pushing all its data into skipforward.net would lead to high load on the system as well as confusion of the users[20]. skipforward.net can search in the full set of DBTropes feature types, telling the bot to push selected feature types to skipforward.net. Communication between skipforward.net and the bot is done via XMPP. In order to keep the subsumption hierarchy of feature types in DBTropes intact, the bot builds a reduced DBTropes feature type hierarchy based on the manually selected feature types, but includes superclasses and neighborhoods of these selected feature types, as well. This is implemented by ad-hoc logic including a spreading activation algorithm.

Trust and Personalized Views

Incremental Updates

Skipforward implements a trust metric based on user similarity on a per feature type basis as outlined in Section 4.3.3. The incremental approach is implemented by the class `AggregatingModelListener` which accumulates any changes occurring in the database of the user. Once there has been no update for some seconds, it triggers an incremental update of the similarity matrices stored in the class `SimiManager`. Matrices are handled by the Colt library[21] and serialized to disk using Kryo[22]. Similarity matrices can always be recalculated from scratch from the RDF data they are based on. On-disk persistence is a speed optimization as building similarity matrices can take minutes on large databases (or days for extreme cases such as the DBTropes demonstrator, see Section 4.5.4). Details about the incremental calculation approach can be found in [Mit08].

[20] There is a separate limited demo installation for DBTropes data in Skipforward though, see Section 4.5.4.

[21] `http://acs.lbl.gov/software/colt/`

[22] `http://code.google.com/p/kryo/`

Metadata Exchange Protocol

XMPP

Skipforward uses XMPP as basis for metadata exchange[23]. Basically, this means that every user runs a Skipforward client that connects to the user's XMPP server. The Skipforward client can request and send metadata using the Skipforward metadata exchange protocol, which, in turn, is based on XMPP. The skipforward.net demo installation includes both a Skipforward installation and an XMPP server. It is possible to log in the Skipforward system running on skipforward.net with any XMPP account, though.

Since provenance of Skipinions ontology instances is encoded as namespace in the URI of the instance (which is, thus, in fact a URL such as *skip://user@host/*), we can request further metadata or updates using the respective namespace. Data exchange between peers is based on full replication, sending compressed RDF/N-TRIPLE diffs whenever changes in peer's data models are detected. This allows simple distributed storage of metadata. Note that this means that, ultimately, no central server is needed.

Linked Data

Skipforward and skipforward.net expose their data according to Linked Data (see Section 2.2) standards[24]. Since there is no centralized data store but only individual user databases that are comprised of the user's and the user's peers' data, one database to expose using Linked Data standards has to be configured. On skipforward.net, this is the database of the *demo@skipforward.net* user who aggregates the data of all other skipforward.net users. Since Skipforward's own *skip://...* URIs are not Linked Data compliant, conversion to Linked Data compliant URIs is done on-the-fly. It has to be noted that in case a user is using an XMPP account that is not one on the skipforward.net XMPP server, his data will not get exposed via Linked Data interfaces since URI host parts do not match.

User Interface

Basic Info Pages

The Skipforward user interface[25] is based on the following principles. The basic types of things users can browse are items, item types, feature types, and users. Accordingly, for each of these things an own type of information page exists, each listing additional information that is useful in context.

[23] XMPP synchronization is implemented as an software layer *JRSync* that accesses the Jena TDB RDF model independently from the rest of Skipforward and can thus get reused in other projects. Its main Java class is `SyncerThread`.

[24] Linked Data handling is done the Java servlet class `LinkedData`.

[25] Java package `de.opendfki.skipforward.ui` contains code building UI views. `de.opendfki.skipforward.ui.web` contains Web/HTML specific code. RSS/HTML templates can be found in directory `/src/template/`.

- The information page about an **item** (see Figure 4.5) lists the item description and reviews; annotations of that item, both individual and aggregated; similar items; annotation recommendations for that item.
- The information page about a **feature type** (see Figure 4.7) lists super- and subclasses of the type; the description of the feature type; example items for that type (items that are annotated by many people with that feature type using applicability 1.0).
- The information page about an **item type** (see Figure 4.3) lists the item type description; super- and subclasses of the type; and instances of the item type.
- A **user** information page (see Figure 4.6) lists the feature types with high or very low user similarity to the current user (the feature types that user typically (dis)agrees about with me) and annotations recommendation targeted to improve the expert recommender.

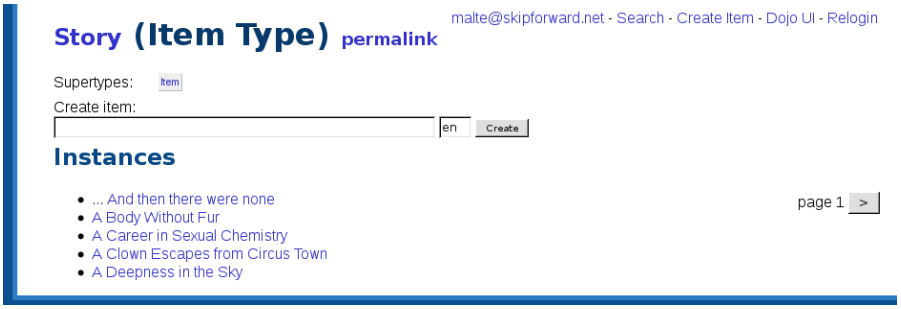


Figure 4.3.: Skipforward item type information page.

Applicability and Confidence

Since applicability and confidence are omnipresent in the system, a compact and intuitive way had to be used for display and input. An icon depiction was chosen: Both applicability and confidence are shown as one icon, a circle, with the circle color denoting applicability, and circle size denoting confidence. This can be seen in Figure 4.5 in the "Missing Features" section of the page. A red circle represents applicability -1, a green circle applicability $+1$, a yellow circle applicability 0. A big circle represents confidence 1.0, a small circle represents confidence 0.1[26]. Whenever applicability and confidence values have to be specified, e.g., when creating a feature instance, such a circle icon is shown as well,

[26] While confidence 0.0 is supported in the backend, since it is not discernible from "no feature present", this is not supported in the user interface.

which can be manipulated by clicking and dragging, or clicking and selecting from a matrix of options.

Mouseover Information

At almost all places, mouseovers are available, showing additional information. For items, their type and textual descriptions are shown. For feature types, their description is shown. For applicability/confidence icons, numerical values and a textual paraphrase of the meaning of this value is shown. Annotation recommenders show the explanation for the recommendation as mouseover, too.

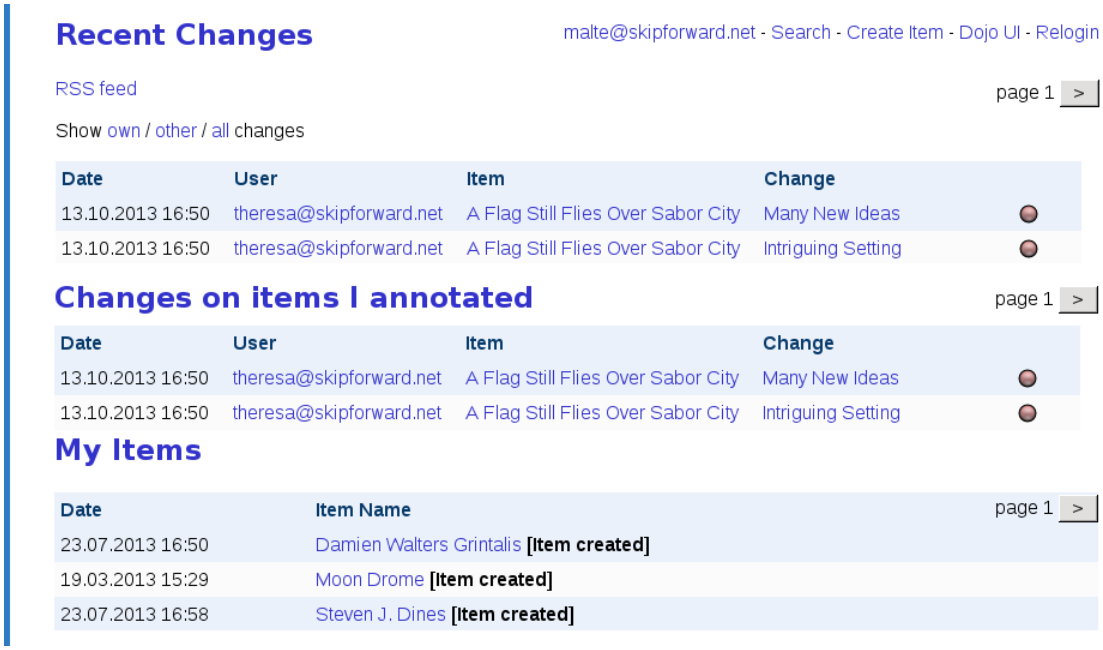


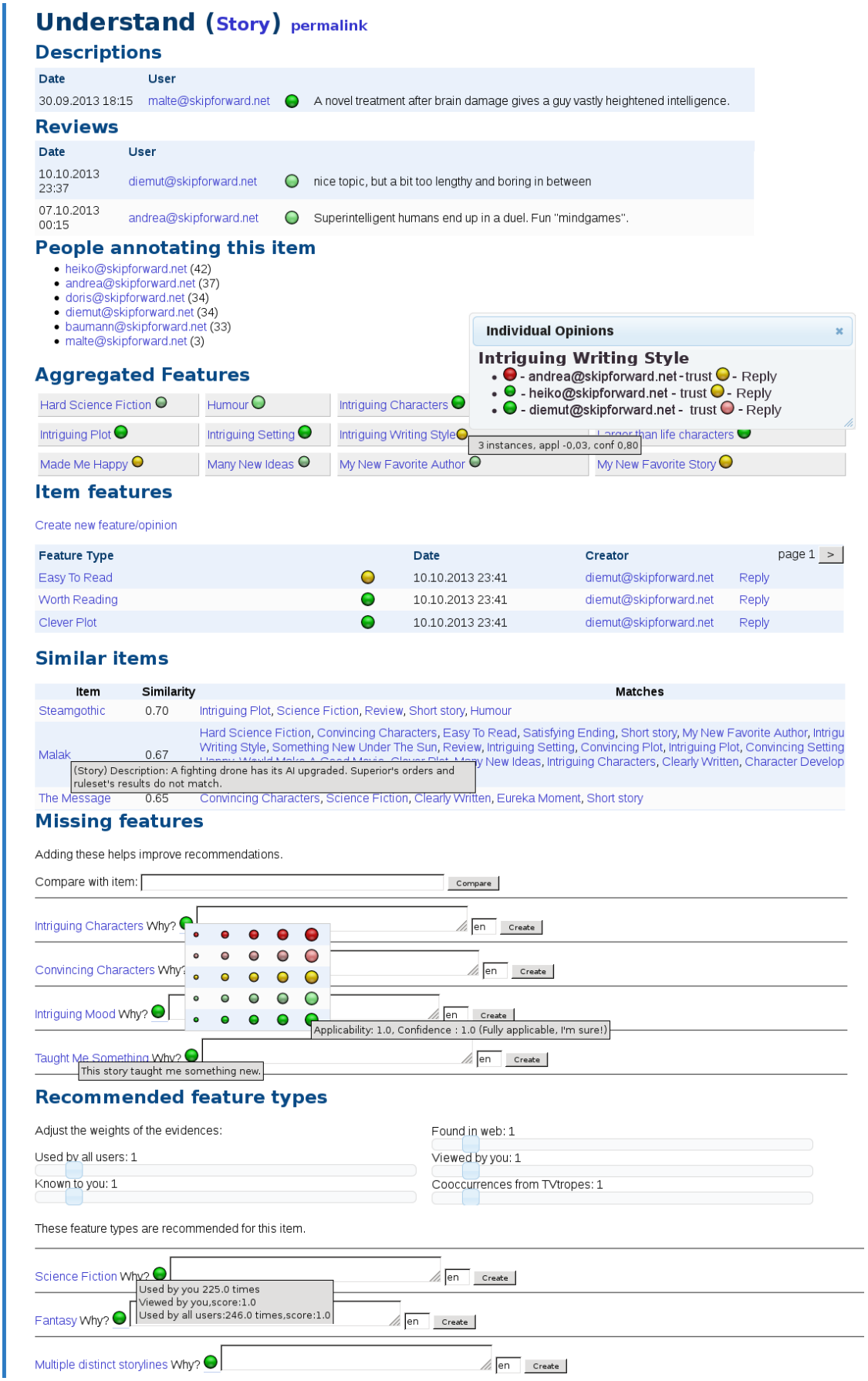
Figure 4.4.: Skipforward landing page.

Landing Page

After logging in into Skipforward, the user is sent to his landing page (Figure 4.4). This lists recent changes in general, recent changes for items the user annotated personally, and the user's items.

Item Information Page

The information page about an individual item (Figure 4.5) is rather complex and warrants a closer description. At the top of the page, any Description and Review annotations for the item are shown in their verbose form. Then, an overview of people that have annotated the item is given, along with the number of annotations every user has created. In the "Aggregated Features" section of the page, all feature types the item has been annotated with are listed, with icon depictions of their aggregated applicability and confidence in that aggregation (for details on user similarity and aggregated views, see Section 4.3.3). By hovering over the icon, individual annotation instances that contribute to the aggregation can be seen. Recent individual non-aggregated annotations are listed



108 **Figure 4.5.: Skipforward item information page with mouseover texts.**

in the "Item Features" section as well. This is followed by the "Similar Items" recommender output which lists items with similar features as the current one, along with explanations for the match. The item information page ends with two annotation recommenders. Recommender functionality gets detailed in the following.

Recommenders

Item Recommender

The item recommender[27] shows items similar to the current item (Figure 4.5). It is based on a similarity metric comparing aggregated features assigned to the two items in question and reuses the backend of the advanced search functionality. Details this can be found the previous section on Skipforward's search functionality.

"Missing Features"

The annotation recommender[28] described in the following tries to improve item recommendation quickly. It implements the first approach described for the *Annotation Recommender: Item* in Section 4.3.7. The recommender shows a list of feature types that the current item has not yet been annotated with. For generating this list of feature types, item recommender output is reused. The annotations present for recommended items are compared with the annotations present for the current item. Any feature types for that a statement exists for the recommended item but not for the current item is presented to the user for annotation. Therefore, annotating using the annotation recommender quickly improves the quality of results given by the item recommender. However, it will not request annotations for feature types for which statements already exist by other users: using it will not improve the expert recommender output. Also, its annotation recommendations ignore some of the context of the item and user, which is exploited by the recommender that is described in the following.

Annotation Recommender: Item

The other three approaches described for the *Annotation Recommender: Item* in Section 4.3.7 are implemented as well[29]. The user interface is separated from the annotation recommender described above for historical reasons. Sliders allow the user to set the weight of each evidence source and rank the annotation recommendations accordingly. For evidence, it supports the following sources:

- "Used by all users" – overall popularity of a feature type, measured by the number of feature type instances.

[27] Java classes `CustomSearch` and `RecommendationAction` handle ranking items according to a preset profile. The similar item recommender code can be found in `ItemInfo`.

[28] The implementation can be found in the Java class `ItemInfo`.

[29] Java package `de.opendfki.skipforward.featuretyperecommendation` contains this recommender's Java code. The code of the Web search evidence source can be found in the `FeatureTypeEvidenceWeb` Java class.

- "Known to you" – feature types used by the current user for annotations.
- "Found in Web" – evidence for occurrence of this feature type has been found in Web search (see below).
- "Viewed by you" – feature types whose info pages have been viewed by the current user.
- "Co-occurrences from TV Tropes" – a static co-occurrence analysis on TV Tropes/DBTropes has been carried out; this evidence source returns co-occurrences to feature types the current item has already been annotated with, based on that analysis.

The Web search evidence source uses the Microsoft search engine `bing.com` for its Web search. The search query is built from the item label and author name, if available. For the first 10 hits, the result documents are fetched and the main text extracted using the boilerpipe library[30]. In the resulting plain text, occurrences of labels of DBTropes feature types are counted. If any label occurs often, this is taken as high evidence for this feature type occurring for the item in question. For speed reasons, fetching HTML documents and searching them for label occurrences is done in parallel threads. A thorough description of this evidence source can be found in [Ami12].

Annotation Recommender: Expert

The *Annotation Recommender: Expert*, as described in Section 4.3.7, aims at recommending annotations that improve the expert recommender's results quickly[31]. In contrast to the annotation recommender discussed previously, this recommender's output are item/feature type pairs. The implementation calculates three matrices whose aggregated output is used to rank annotation recommendations. Two matrices associate scores to item/feature type combinations; one matrix associates a score to a feature type. To reiterate the ranking criteria from Section 4.3.7:

1. Feature types that have been previously annotated by users that are missing correlations with the current user for that feature type are most important.
2. For ties, feature types with no existing correlations between the current user and all the other users annotating items with that feature type should get preferred.

[30] `http://code.google.com/p/boilerpipe/`
[31] Class `RecFeaturesForExpertRec` contains this recommender's Java code.

3. For remaining ties, feature type/item combinations that maximize the number of additional correlations between the current user and other users should get preferred.

Criterion 1 is addressed by the *FeatureMissingCorrelation* matrix. **Criterion 1**

In the following, i is an item, ft is a feature type, F is the set of all annotations, u_c is the current user, u is any other system user, and U the set of all users.

$$FeatureMissingCorrelation(i, ft) = \Big|\{u | u \in U, (u, i, ft) \in F, CoRatings(u, ft) = 0\}\Big|$$

CoRatings is defined using the two following functions:

$$Common(u, ft, i) = \begin{cases} 1 & \text{If both the current user } u_c \text{ and user } u \text{ annotated} \\ & \text{item } i \text{ with feature type } ft \\ 0 & \text{Otherwise} \end{cases}$$

$$CoRatings(u, ft) = \sum_{i \in Item_c} Common(u, ft, i)$$

Thus, $CoRatings(u, ft)$ is zero in case there are no items for which both the current user and u have created an annotation of the type ft.

$FeatureMissingCorrelation(i, ft)$ represents the number of users who have created an annotation of type ft for item i, and the current user has no correlation with these users with regard to ft at all.

Criterion 2 is addressed by the *FeatureTypeMissingCorrelation* vector. **Criterion 2**

$$FeatureTypeMissingCorrelation(ft) = \begin{cases} 1 & \text{If } \sum_{u \in U \setminus u_c} CoRatings(u, ft) = 0 \\ 0 & \text{Otherwise} \end{cases}$$

$FeatureTypeMissingCorrelation(ft)$ is one in case there is no correlation to any other user with regard to ft.

Criterion 3 is addressed by the *Overlap* matrix. **Criterion 3**

$$Overlap(i, ft) = \sum_{u \in U_i} \frac{1}{2^{CoRatings(u, ft)}}$$

with U_i denoting the set of all users for who an annotation of item i with type ft exists.

From the *Overlap* matrix (which contains fractional numbers), the matrix *OverlapRank* is built by sorting the numbers, and substituting their sorting rank, which is an integer.
$OverlapRank(i, ft)$ is low in case many users who have annotated item i have many correlations with the current user with regard to ft.

Aggregating Matrices

Finally, the three matrices get aggregated.

$$\begin{aligned} Score(i, ft) =& w_{maxMissing} * w_{maxOverlap} * FeatureMissingCorrelation(ft) \\ &+ w_{maxOverlap} * FeatureTypeMissingCorrelation(i, ft) \\ &+ OverlapRank(i, ft) \end{aligned}$$

$w_{maxOverlap}$: the maximum value found in the $OverlapRank$ matrix $+1$
$w_{maxMissing}$: the maximum value found in the $FeatureMissingCorrelation-Rank$ matrix $+1$

The user interface of this recommender can be found on the user information page (Figure 4.6). At the bottom of the page, items and feature types of this recommender get listed. A thorough description of this recommender's approach and implementation can be found in [Ami12].

Expert Recommender

In Figure 4.7, a feature type information page is shown, including the expert recommender[32]. The expert recommender works on a per-feature type basis, recommending users who expressed similar opinions concerning a selected Skipforward feature type compared with the current user ($sim_t(u_x, u_y)$ in Formula 4.1).

Search

Search

Skipforward supports simple subtext search for item (type) and feature type labels. Most interesting, though, are its advanced search capabilities (see Figure 4.8)[33]. Advanced search works by first *filtering* all items based on user-selected criteria, then *ranking* the found items according to selected other criteria. Filtering can be done based on the following information:

[32]The expert recommender Java code for the UI frontend can be found in `FeatureTypeInfo`; user similarity calculation is done in `SimiManager`.
[33]In Java package `de.opendfki.skipforward.search`, the search backend can be found.

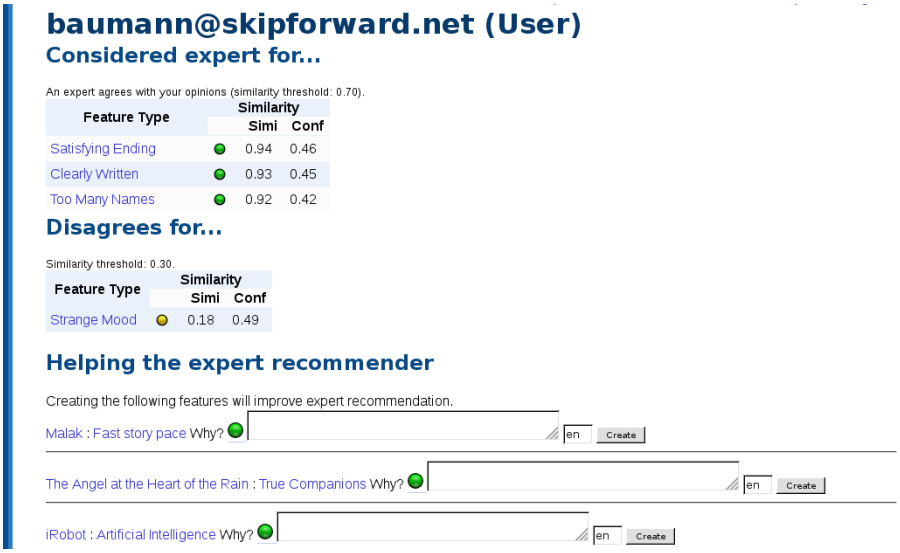


Figure 4.6.: Skipforward user information.

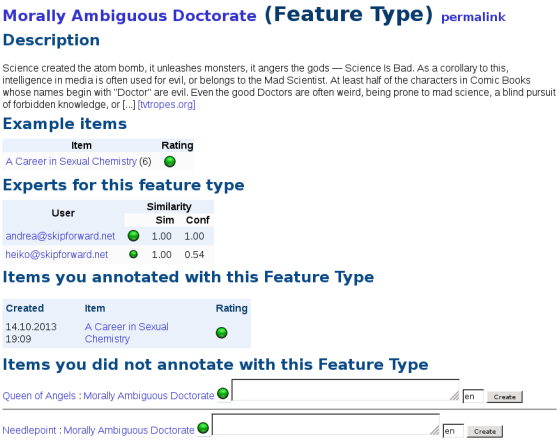


Figure 4.7.: Skipforward feature type information page.

- Find items based on their type.
- Find items based on annotations. This can be by aggregated features for one feature type or by a specific feature instance by some user.
- Find items by their connections to other items: for example, searching for books written by a specific author can be done this way.
- Find items that have been annotated by a specific user.

The search *filters* filter all items by the criteria given, in the order given in the interface; only items matching these criteria are listed as search results. This is

why feature type filters support *ranges*, e.g. "Show items with item type Story annotated with feature type *Action Girl* and applicability 0.7 to 1.0". *Ranking* criteria (for which only feature types are supported) need fixed values, e.g., "Rank items with feature type *Science Fiction* and applicability 1.0 highest". In contrast to filters, the order of ranking criteria in the UI is of no consequence; instead, they are given a *ranking weight*. In the results list, matching feature types for ranking criteria get shown for each item as an explanation for its ranking. This is especially helpful if many ranking criteria are specified. The same functionality is visible for similar items listed on item information pages (Figure 4.5). In fact, the same backend is used there: The items listed as similar items are the result of an advanced search with the features of the current item given as search ranking criteria.

Explanations

RSS feeds

An extra functionality of advanced search is that search criteria can be saved and then appear under the "My Searches" tab. There, RSS feed links are available that allow the user to subscribe to individual search tasks. This way, users can easily stay up to date in areas of their interest without having to look at all recent changes.

Example

In Figure 4.8, a search for Short Stories that *malte@skipforward.net* annotated with the *Entertaining Humor* feature type is shown. The results are ordered by their aggregated *Review* score.

4.5. System Use Case Examples

Over time, Skipforward and the demonstration installation skipforward.net have been used for several different domains. Some of these use cases are listed in the following.

4.5.1. Books/Stories Domain

The books/stories domain has been the longest-running thematic domain on skipforward.net. The *storyteller* ontology (an ontology containing some basic feature and item types for the domain such as Story, Author, and *written_by*) has been created in 2008. Since then, about 300 story instances have been created and annotated. About 160 stories of these are short stories published in the bimonthly Interzone magazine. For annotation, apart from the small domain ontology that is part of the storyteller ontology, since 2009 DBTropes (see Section 5.7) has been used additionally. Most of the screenshots in the figures of this chapter are based on this domain.

Figure 4.8.: Skipforward advanced search.

In the 2013 evaluation, a subset of this domain was used. This gets presented in detail in Chapter 6.

4.5.2. Board Games Domain

This started as an import of *Brettspielbrowser* (BSB) dataset created by Sven Schwarz. BSB was a standalone Web-based database with an advanced UI[34]. However, its multi user features were lacking, and since the Skipforward data model was designed with the BSB data in mind, the data was imported (Fig-

[34] `https://web.archive.org/web/20040319053018/http://brettspielbrowser.de/Alle_Spiele.html` – archived version of 2004

ure 4.9). The dataset contains about 150 board game feature types and descriptions of about 300 board games that consist of about 1,800 feature instances. This domain serves as a good demonstrator for the similar item recommender and the "Missing Features" annotation recommender.

Figure 4.9.: Skipforward board game example.

4.5.3. Music Domain

Music Genome Project

Skipforward's initial focus was the musics domain. In [Mit08], an importer of MP3 metadata was written, and the evaluation of the incremental user similarity approach developed in the thesis was carried out on song data. For annotation, primarily feature types inspired by the Pandora recommender service respectively the Music Genome Project[35] were available; among the 659 feature types are types such as *Punk Influences*, *Trumpet Solo*, and *Female Vocal*. About 9,000 feature instances of these feature types have been created over time.

4.5.4. DBTropes/TVTropes.org Demonstrator

With the availability of the DBTropes wrapper (see Section 5.7), a lot of data from the story and gaming domain became available. Importing all the data into skipforward.net was neither possible nor practical, unfortunately: The prototype was not able to handle 26,000 feature types, 50,000 items, and 3,000,000 feature instances due to various reasons: for example, initial feature inference takes very

[35] `http://www.pandora.com/`

long on this large dataset; the full replication approach of the data synchronization layer would have meant duplicating DBTropes *user count* times on skipforward.net; also, most recommenders have been implemented in a straightforward way, not doing any offline calculations, and often having $O(n^2)$ complexity with regard to item and/or feature type count, since optimization in this regard was out of the focus of the research. Lastly, the massive amount of data would likely confuse users (for example, by introducing noise in autocompletion UIs).

Simplifying Functionality

In order to create a separate demonstrator instance of the system that runs with the DBTropes data exclusively[36] (Figure 4.10), some functionality had to be simplified[37]:

- most UI list views in that demonstrator present only unsorted lists
- the item recommender uses a Monte Carlo approach for reducing runtime in online calculations, giving non-deterministic results
- multi user functionality is not needed (as DBTropes knows only one user) and has been disabled; no personalized views or trust metrics are available
- the system is read only

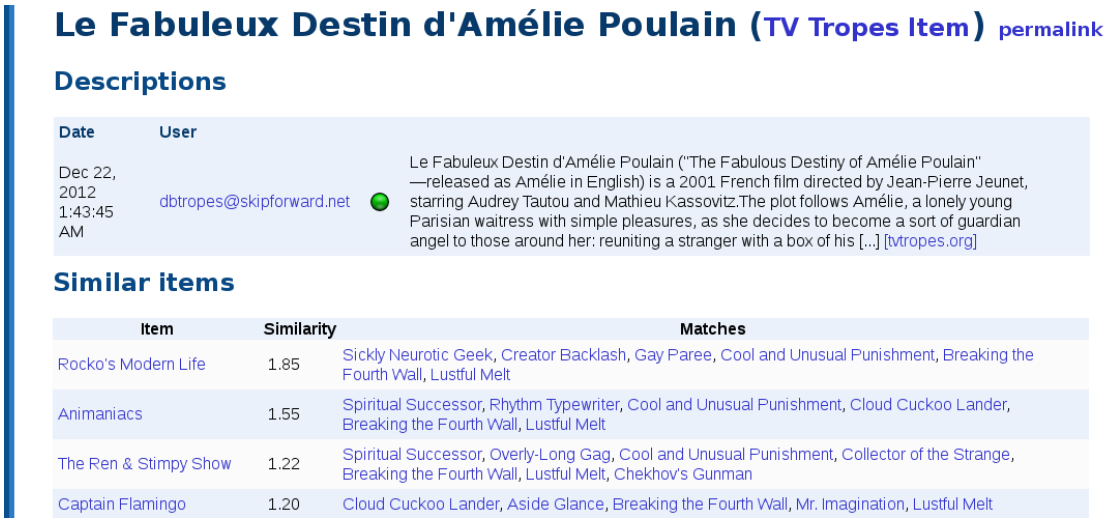


Figure 4.10.: Skipforward DBTropes demonstrator.

New Features

However, new features have been introduced as well: The item recommender uses predefined weightings for feature types/tropes in order to improve item

[36] `http://dbt.skipforward.net/`

[37] The shortcuts can be found in the source by looking for the `BIG_DATA_HACKS` flag.

recommendations. Since in TVTropes items are typically annotated with a lot of obscure tropes, this was necessary to weigh the more important tropes higher in recommendation; otherwise, the "defining" tropes of an item would be lost in the flood of matching minor tropes. For calculating the weights, a TF/IDF approach was chosen. While this leaves room for improvement, it was a pragmatic approach to the problem.

Result

In the end, the `dbt.skipforward.net` demonstrator is an interesting showcase for DBTropes and Skipforward's item recommendation and search/browsing functionality; again, it has to be noted that Skipforward's most interesting feature, its handling of subjectivity and personalized views, is not visible in the DBTropes-based demonstrator. Future work could be extending DBTropes to extract provenance for any statements extracted from TV Tropes. This would require looking at TV Tropes Wiki page histories and elaborate matching of page changes with Wiki users though which is most likely very error prone. Also, some of the simplified functionality probably could be re-added by improving algorithms and using more caching or offline calculation. However, this was out of scope for this thesis.

4.6. Conclusion

In this chapter, the foundations of a novel approach to a resource-centric annotation system are shown. The *Skipforward* approach features clear semantics in its data model and allows users to express negated opinions which allows them to express dissent with the statements of other users. Based on the facts captured in the data model, user similarity is computed, and trust values get derived from this. Trust, in turn, is used to weigh statements of other users, which influences how individual statements get aggregated. In contrast to other similar systems, Skipforward handles trust not on a general per-user basis, but rather calculates trust values for each feature type. It can handle people sharing views concerning one type of feature (e.g., liking the same type of humor), but differing on other levels (e.g., disagreeing concerning liking of characters in a story). This implements personalized views on a fine-grained level, and facilitates unique features such as recommenders that can find users sharing the same opinions concerning specific issues.

Handling Dissent

Trust

Personalized Views

Semantic Search

Since Skipforward has a detailed model of the items contained in its database, it allows the user to do specific semantic search. Additionally, for all its recommender functionality, detailed explanations are available thanks to its hybrid recommender approach. These kinds of detailed explanations cannot be provided in typical collaborative filtering-based approaches.

Explanations

CHAPTER 5
Using External Ontologies For Annotation

5.1. Motivation

The Role of Annotations

Annotations, as discussed in the previous chapters, are key to making content more available, and for offering semantic services (e.g., recommenders and advanced search) that go beyond simple approaches (e.g., text search). However, if these annotations are not machine-interpretable, merely a shift of the problems inherent to machine processing of content has happened: Instead of the resource content (e.g., text or images) that is mostly opaque to machine interpretation, now the annotations are available, which are just as opaque. By providing rich semantic annotations, this problem can be overcome. Semantics, in the context of annotation ontologies, is implemented by information that helps interpreting the annotations. This information can be hierarchies in the annotation classes, synonym/antonym relationships of annotation classes, and other meta-information that helps constructing relations between concepts and instances (details on this can be found in Section 2.1.2).

Semantics

Ontology Re-Use

Building these ontologies and specifying relationships between classes is work intensive. Starting an ontology from scratch is often counter-productive as well, since defining an own vocabulary makes it incompatible (or *incomprehensible*) to other services in the same domain—that is, unless further steps are taken to *align* the ontologies and specify links between matching classes. In short: Ontology *re-use* is needed.

Human-Readable, but not Machine-Interpretable

Unfortunately, while ontologies and vocabularies for many domains exist in various forms[1], they are often part of services and not available (externally) in machine-readable form. For example, item categories and hierarchies in Web shops (e.g., household items, photo equipment, computer parts) are typically present in backend databases; however, only HTML is exposed to the Web, which is suitable for presentation to humans, but in general not machine-interpretable. Other examples are Wiki-based community efforts such as Wikipedia or TVTropes.org. Domain information including categories, hierarchies, and cross-references is available in these sources, but it is not available

[1]These can be available both on the Web and offline, but the focus here is resources that are available online.

in a truly machine-readable format. Such a machine-readable format is RDF (see Section 2.2); the Linked Open Data approach is a way of making RDF available on the Web (see Section 2.2).

Reasons For Closedness of Web Services

There are several reasons for existing services focusing on human-readable representations of their data and ontologies exclusively. These are historical, technical, social, and economical reasons:

- Until recently, semantic technologies, and online APIs in general, were not in widespread use. With no consumers of exposed data, there was no incentive to provide data.
- Many services (in terms of their underlying software as well as the community and infrastructure surrounding them) are very complex. Adding Linked Data support can be perceived as an additional unnecessary burden.
- Exposing data as high quality Linked Data can be difficult. In contrast to preparing data for human consumption, data intended for machine processing should be as clean and sound as possible. This might require data cleaning and possibly additional meta-data.
- The data structures underlying services are often not centered around semantics at all: for example, typical Wiki-driven Websites use sets of text documents in their backend. Exposing this as high quality Linked Data requires additional nontrivial processing steps[2].
- Services and community Websites have their own priorities. Making their data available for outside users likely is not a priority, and very likely is not at the core of the skill set of the maintainers.
- Commercial services typically base their business model on treating their data as a company secret.

Licenses

In case the data is available under an appropriate license[3], *wrapping* the data into a service separate from the original Website might be possible. Keeping the Linked Data service separate from the main data source can be beneficial due to separation of concerns: Adjustments and transitions concerning the Linked

[2] Just exposing the set of documents as Linked Data would be easy but defies the point, as this data is not machine-interpretable either.

[3] cf. GNU Free Documentation License (GFDL) `http://www.gnu.org/copyleft/fdl.html` or Creative Commons (CC) licenses `http://creativecommons.org/` – for a thorough discussion of licenses suitable for data, see [MSH08].

Data output can be carried out independently from the original data and software, not confusing the original community or complicating the work of the software/installation maintainers. Also, specialized communities can form: Linked Data maintainers typically need quite different skills and focus than the original community.

Different approaches exist for wrapping services to provide Linked Data. This chapter discusses possible solutions and explores one specific approach in more detail. Also, special consideration will be given to ontology re-use and interlinking instances of different data sources to enrich the data available.

5.2. Conversion Approaches

Multiple approaches exist for making existing structured or unstructured data available as Linked Data. Figure 5.1 shows a few of these approaches. In the following, an overview of these approaches is given.

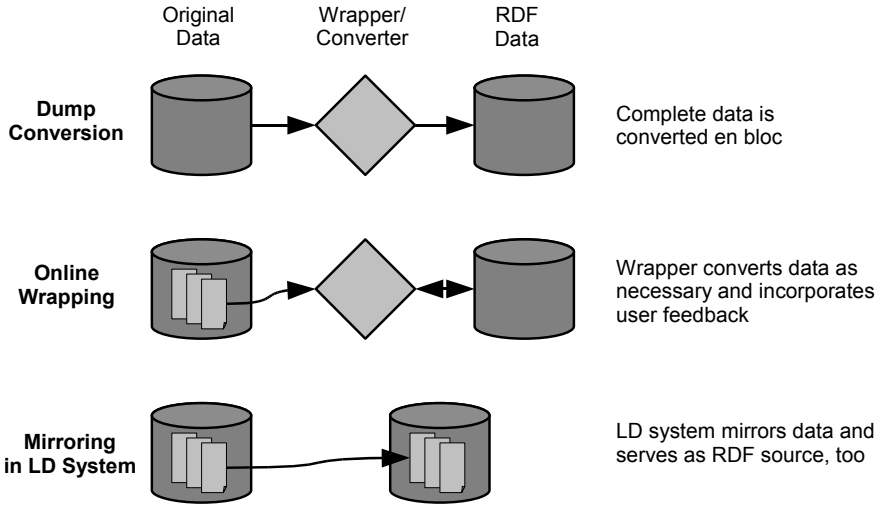


Figure 5.1.: Three wrapping approaches.

5.2.1. Dump Conversion

Dump conversion is essentially a one-shot (but possibly multi-pass) conversion of a data collection (in HTML or other formats) to RDF, then exposing this (static) RDF in the Web using a simple Linked Data service. This is the approach chosen

by DBpedia[4] [LIJ+14]. DBpedia takes the database/HTML dumps provided by Wikipedia, and uses information extraction techniques to extract structured data from the Wikipedia info-boxes. This structured data is represented in RDF and exposed as Linked Data.

Easy To Set Up

In case the actual mapping of the data to RDF is straightforward, dump conversion is easy to set up. A global view of the data to transform is present and no incremental changes in the data need to be taken care of. Existing conversion and extraction tools packaged as file-based command line tools can get reused directly. Dump conversion's technical merits lead to some structural problems with the approach. For example, it is difficult to correct extracted data in a timely fashion: Consider a user finds an error in the Linked Data output and fixes the underlying problem in the original source, or adds hints to the conversion routine (in case the conversion mechanism supports user feedback). Still, the immediate benefit of reporting bugs is not there: Changes in the converted data will be reflected in the RDF only after the next conversion run. Additionally, this long roundtrip time makes fixing complex errors difficult for the user, as the first attempt at fixing the problem might not succeed, and every attempt to fix the problem takes a long time. This discourages users from contributing or leads to problems with colliding community priorities[5].

High Turnaround Time

Provisions can be taken to allow incremental updates on the extracted data—processing only pages in the input data that are marked as having changed since the last run. This leads us to the next wrapping approach.

5.2.2. Online Conversion

As opposed to bulk conversion, *Online conversion* means the original data is only converted to RDF once information about that particular Linked Data resource is requested. Depending on the domain, the approach can be simple when data dependency between extracted resources is low and scale and latency requirements are modest. On the other hand, the approach can get complicated in case of complex data dependencies, or high performance requirements. If the wrapped data is basically a set of instances of a fixed ontology class, then almost no data dependency between resources exist, and each page can be treated independently. If, however, the wrapped data includes both instances *and* the data's schema, then changes to schema resources may invalidate large portions of the previous wrapping output. Thus, for online wrapping of nontrivial do-

[4] `http://dbpedia.org/`

[5] This can be seen in DBpedia: DBpedia/Linked Data people want infoboxes to be more consistent which is not a priority for the Wikipedia community.

mains, dependency management is needed.

APIs as Data Sources

In case the original data source provides Web APIs, a Linked Data service can be built that wraps these APIs online and exposes the data as Linked Data. Depending on the domain, such a wrapping service can integrate information retrieved from many sources. See [BCG07] for a simple example of such an approach.

User Feedback

One benefit of online conversion is that user feedback can be included in the wrapper's output quickly. User feedback can take on the form of hints and corrections as well as additional information not present in the original data. For collecting user feedback, the wrapper needs to provide an own user frontend.

Incorporating additional data (e.g., links to other relevant Linked Data resources) can be done seamlessly. Last but not least, the extracted data is easy to keep up-to-date. Several optimizations for online conversion are possible—this will be explained in more detail in the next section.

5.2.3. Mirroring in a Linked Data-Enabled System

For systems used for content or data management such as Wikis and databases, variants of the systems exist that support Linked Data. For Wikis, this is the family of *Semantic Wikis* (see Section 2.4.2). Mirroring a service's data in such a Linked Data-enabled system is another option for exposing the original data as Linked Data: for example, for Wikipedia this could be implemented by mirroring (and extending) its content in a Semantic MediaWiki [KVV06] installation. As a bonus, the semantically enabled systems typically provide additional functionality that may be of direct interest in the application domain. Also, fixing extraction problems manually is easy.

Synchronizing

However, a number of problems arise in this approach: Imagine the Linked Data-enabled system containing a full copy of the original data. Typically, for enabling all functionality of the Linked Data-enabled system, additional data has to be added, or subtle changes to the existing data must be done. In the (Semantic) MediaWiki example, markup must be changed, e.g., links between pages must get typed, and categories adapted. This represents lots of small changes scattered all over the existing data, rendering automatic synchronization with the original data source nearly impossible. Finally, for large systems and data amounts, it is unclear whether the Linked Data-enabled systems and their additional functionality scale sufficiently.

The focus of this chapter lies on online wrappers. The components of a typical online wrapper are described in the following.

5.3. Components of an Online Wrapper

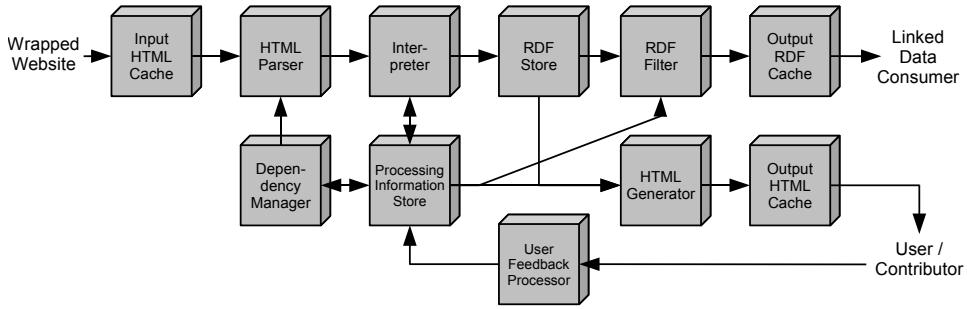


Figure 5.2.: The components of the online wrapper. Arrows show information flow. Caching management not shown [KG10].

Data Flow

In Figure 5.2, the components of an online wrapper are shown along with the data flow between them. The *input HTML cache* helps relieving the wrapped Website from unnecessary load. The *HTML parser* extracts interesting information from the fetched HTML pages and outputs data in an ad-hoc data format (syntactically only). XPath[6] expressions or general screen scraping techniques can be employed in this step (cf. *Piggy Bank* [HMK07], an RDF screen scraping framework). The *interpreter* generates RDF from these data snippets. Typically, additional information for generating RDF is needed—this information is fetched and updated in the *processing information store*. The *dependency manager* controls updating wrapped pages in case metadata/processing information data changes. The *RDF filter* hides RDF statements marked as invalid by user feedback; not pushing these statements to the RDF store from the interpreter is impractical since the HTML generator needs to know about the exact statements in order to allow unhiding them.

5.4. Consistency Management and Caching

Once a Linked Data resource is requested the first time, the online wrapper fetches the associated original HTML resource, parses it, and stores the resulting RDF to a triple store. However, parsing the HTML is not always straightforward. Depending on the page type and further information, interpretation of the HTML data may differ. Consistency issues also arise when updating processing hints (e.g., changing the type of a page in the online interface). In this

[6] `http://www.w3.org/TR/xpath`

case, rulesets derived from the underlying RDF schema can be used to determine resources that depend on the current resource and that need to get updated.

Levels of Caching

To improve performance and reduce load on the underlying service, caching has been used at several levels of the processing chain:

1. The original HTML pages need to get cached in order to minimize the load the wrapper imposes on the underlying service.
2. RDF output needs to get cached in order to speed up Linked Data service response times. If there is the need for resource info on a resource's wrapper HTML page that should not be visible in the Linked Data output (e.g., manually hidden incorrect statements), caching RDF is necessary on two levels.
3. The HTML describing each resource and providing the means for user feedback should be cached[7].

5.5. Fetching Updates

Update Strategies

With caching explained in the previous section, the question of when to re-fetch information from the wrapped data source in order to account for potential updates arises. Several solutions for this exist.

- *Data expiration*: Each resource is valid for a pre-set time and re-fetched once that time expired and the resource gets accessed.
- *Continuous updates*: In the wrapper, a background process keeps re-fetching resources from the wrapped site continuously.
- *Push updates*: In case the wrapped Website features a change feed in the form of an RSS feed or similar, this can be used to trigger updates in the wrapper.

When exposing data in the Linked Data Web, the exposed data should be matched with their equivalent counterparts in other existing Linked Data sources. Otherwise, the data is not actually *Linked* Data, and finding out whether two data sources (with their different URIs/URI namespaces) are about the same resources or not becomes a (hard) problem. In the following, the process of finding instance matches and its challenges are outlined.

[7] These pages easily grow to several hundreds of kilobytes.

5.6. Interlinking With Other Linked Data Sources

Why Interlinking With Other Sources?

For truly providing Linked Data, not only re-use of vocabulary (i.e., re-use of RDFS properties and classes of established ontologies) is crucial, but also interlinking of instances is necessary. For example, if the data source that is to be wrapped contains occurrences of well-known people, interlinking with existing Linked Data sources that mention these people as well (e.g., DBpedia) is advisable as (i) any Linked Data consumer will be able to profit from the merged knowledge of both data sources, and (ii) there is less ambiguity whether two resources on the data sources are the same or not. Another common resource type that is worth interlinking with is geographical data; in this case, GeoNames.org is a candidate Linked Data source. Technically, this interlinking is implemented by creating an RDF triple using the *owl:sameAs* property with the two matching resources as subject and object.

In case no interlinks are present in the original data, gathering evidence for resource matches must be done first. The following points need to be taken into consideration if generating matches automatically.

5.6.1. Inverse Functional Properties

In case *inverse functional properties* or *primary keys* such as ISBN numbers (for books) or other identifiers that uniquely identify a resource are shared by both data sources, interlinks can be generated with very high confidence easily.

5.6.2. Similarity Metrics

In case no inverse functional properties exist, similarity metrics on the resources's properties can be used to derive evidence for semantic matches. This is a process that typically requires a lot of manual work and tweaking as well as a domain-specific ruleset: What properties to consider for comparison? How to translate literal representations from one source to the other (e.g., different units for lengths)? What thresholds to apply for a "perfect match"?

5.6.3. Type Compatibility

One special subproblem when applying similarity metrics to resources of two different data sources is what resources to apply the metrics to actually. Limiting the pairwise comparison to resources of types that can be compatible is reasonable. However, since the class hierarchies of both data sources will probably

be different (e.g., one might talk about "films", the other about "movies"), this is not as straightforward as it seems. Essentially, before instance matching, the two ontology hierarchies need to get mapped. This presents its own problems, e.g., mismatches in design decisions with regard to class/instance levels[8].

5.6.4. Granularity

Differing resource granularity means that different data sources model instances on different levels of specificity: for example, in the movie domain, one source contains one resource representing the "Batman" franchise, whereas the other source has one resource each per movie and comic. A specific problem here is that concerning the instance similarity metrics all these resources are likely to seem very similar—confusing a book with the movie that is based on the book is very easy when looking at the metrics only. To avoid these types of mistakes, type compatibility must be checked thoroughly, and rules have to be set up that make sure as few false positives as possible are generated.

Type Granularity

5.6.5. Recall vs. Precision

As usual in the Semantic Web field, for interlinking of data sources, high Precision is more important than high Recall. Lacking Recall in instance matching leads to (i) fewer results in retrieval (if resource properties cannot be found since aggregation of resources has not taken place) or (ii) duplicates in retrieval results (when resources are found from different sources that actually are semantically identical). Bad Precision, instead, leads to (i) completely wrong results in retrieval (up to retrieval results that do not even match the type of the actual search), and, in combination with inference, (ii) almost completely unpredictable behavior.

Recall

Precision

In the following, an example for the online wrapping approach is given.

5.7. Prototype Implementation: DBTropes

The DBTropes.org Linked Data source[9] is the result of applying the online wrapper approach (see Section 5.2.2) to the TV Tropes[10] community Wiki. DBTropes

[8] Imagine one source modeling countries as classes, the other modeling countries as instances.
[9] `http://dbtropes.org/`
[10] `http://tvtropes.org/`

features a Web frontend that allows users to tweak the way resources are processed, removing incorrectly extracted facts and helping to link DBTropes resources to the rest of the Web of Data (cf. Section 5.6).

5.7.1. The TV Tropes Wiki

TV Tropes is a community-maintained Wiki containing a catalog of *tricks of the trade* for writing fiction, known as *tropes*. According to its introduction page:

> *"Tropes are devices and conventions that a writer can reasonably rely on as being present in the audience members' minds and expectations."*

Typical Page Structure

The Wiki includes tropes[11] ranging from well-known plot devices such as Deus Ex Machina[12], to the elaborate and complex, yet easily escapable, contraptions James Bond villains will employ to execute the hero[13], or the often naive portrayal of computer science in fiction[14]. Each page of a trope contains a description as well as links to related tropes and links to examples in which this trope occurs, almost always with a comment explaining why that trope is relevant in the context of the linked example. In turn, each item page contains an item description, as well as commented links to the tropes occurring in the described item. Tropes might not just occur but also be *averted*[15] (a situation to enact the trope is set up but—perhaps due to the writer not knowing the trope—the trope is not played) or *subverted*[16] (the writer sets up a situation to play a trope but then introduces a sudden twist). As its name implies, TV Tropes was initially concerned with television shows, but the scope has since expanded to cover all sorts of fiction from movies and books, to computer games and comics.

[11] As of mid-2013, TV Tropes includes about 30,000 trope pages, and almost 50,000 item pages.
[12] `http://tvtropes.org/pmwiki/pmwiki.php/Main/DeusExMachina`
[13] `http://tvtropes.org/pmwiki/pmwiki.php/Main/WhyDontYouJustShootHim`
[14] `http://tvtropes.org/pmwiki/pmwiki.php/Main/BeepingComputers`
[15] `http://tvtropes.org/pmwiki/pmwiki.php/Main/AvertedTrope`
[16] `http://tvtropes.org/pmwiki/pmwiki.php/Main/SubvertedTrope`

Focus on Informal Knowledge

TV Tropes does not attempt to create an objectively correct information source like Wikipedia. TV Tropes does not contain or re-model typical factual information about films and other items (e.g., when a movie was created, who directed it, what actors participated). Instead, it focuses on the plot devices employed and the type of characterizations made—which makes the information contained in TV Tropes a perfect addition to the mostly factual data that is already present in the Web of Data. Only with TV Tropes data, finding answers to questions such as *What Bruce Willis movies are there featuring the hero walking away from an explosion without flinching?* becomes possible – a question that needs the combined knowledge of Wikipedia/DBpedia *and* TV Tropes. This was the motivation for creating the DBTropes.org wrapper.

5.7.2. The Ontologies Used

Converting TV Tropes to RDF data was also motivated by being able to re-use the tropes hierarchy and movie annotations within the context of the Skipforward project (cf. Section 4.4). The structure of the TV Tropes Wiki matches almost perfectly the data model described by Skipforward's main ontology *Skipinions* that declares item and feature types[17]. Truth and confidence values allow modeling averted and subverted tropes from TV Tropes. Feature instance comments are taken from TV Tropes directly. Since extracting provenance data from TV Tropes (as in "What user wrote the passage associating trope X with movie Y?") is very hard to do automatically[18], the personalization feature of Skipinions remains unused here.

Visualization in Skipforward

In Figure 5.3, a small part of the data made available by DBTropes is shown, as visualized in Skipforward[19]. On the left side, items are shown; the upper right side displays available tropes (or *feature types* in the Skipforward terminology) along with colored circles if instances for these feature types are present for the selected item[20]; the lower right pane shows instances of the selected feature type (i.e., users expressing opinions about one item with regard to one feature type)[21].

[17] The near-perfect match of the *Skipinions* data model and the implicit data model of TV Tropes can also be taken as an indication that hypothesis H1 holds.

[18] This would require looking at the Wiki page history and trying to match edits to trope occurrences, which in a scenario with anonymous Wiki edits and frequent refactoring—as is the case for TV Tropes—is far from trivial and quite error-prone.

[19] Note the old Dojo-based version of the Skipforward UI is shown; it was dropped for a more verbose but less click intensive UI variant.

[20] The first circle shows the weighted average of applicability of the feature type with user opinions weighted according to their trust value.

[21] Note that here, two users have assigned conflicting opinions about the feature type/trope *A God Am I* to the movie *A Nightmare on Elm Street*. This is for demonstration purposes only;

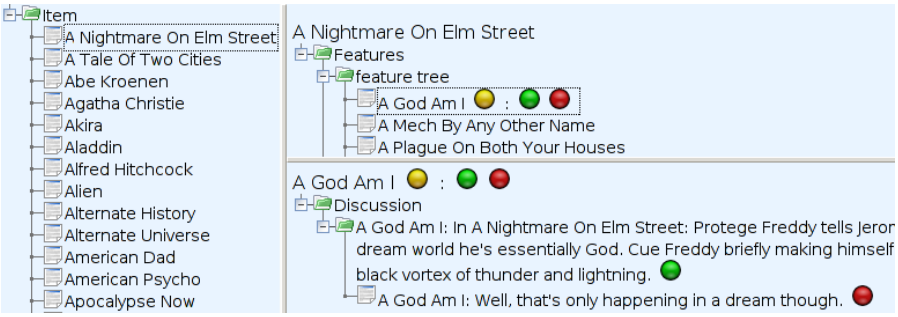


Figure 5.3.: DBTropes data in the Skipforward Dojo-based user interface.

Currently, the tropes hierarchy is present as a Skipinions feature type hierarchy that is represented using *rdfs:subClassOf*. Representing it alternatively as SKOS concepts [MB09] is possible as well.

Ontologies Used

Additionally to Skipinions, parts of other ontologies are used, too: For associating an extracted resource with the original TV Tropes page describing it, *foaf:page* from the FOAF vocabulary [BM07] is used. *owl:sameAs* from the OWL vocabulary [MPPS09] is used for modeling redirects in the TV Tropes Wiki as well as linking equivalent resources of other Linked Data sources. For modeling provenance, copyright information, and basic statistics, the voiD vocabulary [ACHZ09] is used.

Some additional properties are needed for the wrapper, declared in the wrapper's own ontology: for example, the date and time the underlying TV Tropes page was fetched is represented as well as the date and time the HTML page was parsed—these two timestamps may differ due to HTML caching. Finally, processing notes get represented using DBTropes specific properties as well.

5.7.3. (Domain) Challenges When Wrapping

Unfortunately, parsing the TV Tropes Wiki is not straightforward in some regards. Some problems are easy to deal with; others are very hard.

Page Types

One challenge was that item and trope pages were difficult to tell apart automatically: There was no clear indication in the HTML that marks items (movies, games, ...) or tropes. Even to the human reader telling whether a page describes an item or a trope was difficult occasionally. Since this decides the type of the RDF resource generated for a page, this information had to be inferred somehow. Originally, this was solved by a simple inferencing heuristic that kept track of page references, based on the assumption that an item page links to

DBTropes is seen from Skipforward only as one user.

more trope pages than other item pages. This guess could also be corrected manually in the Web frontend if necessary. Fortunately, in 2009 the TV Tropes community (re-)added a feature to mark pages as tropes or items directly in the Wiki.

Hierarchies

Another ontological wrapping problem is extracting proper item and feature hierarchies (i.e., Movie *subclassOf* Item, or RuleOfCool *subclassOf* BiggerIsBetter). Ideally, one would like to have a strict *is-a* hierarchy to keep in line with RDFS subclass semantics. However, such information is not present in TV Tropes. There *are* categories for tropes and items[22]. Unfortunately (for this use case), these are used in a lenient way, enhancing navigation in the Wiki in the first place. This leads to two problems:

1. Categories are often index pages, grouping tropes and items. This typically represents a part-of relationship, not an is-a relationship, with no easy way to tell both apart.

2. Loops are common: for example, pageA *category* pageB *category* pageA. If naively modeling category links as subclass relationships, this would imply identity of pageA and pageB.

Skipforward DBTropes Bot

This problem has not been solved completely yet. For usage in Skipforward, a separate bot that prepares DBTropes data for Skipforward deals with this. The bot gathers evidence for subclassOf relationships (e.g., category links), builds a trope hierarchy from these relationships, and detects and eliminates loops by dropping *subclassOf* statements if necessary.

Concerning hierarchies, similar problems exist for DBpedia. Some recent work has been done that investigates building hierarchies from categories, partially supported by text analysis. See [MMB+13, DI08] and the DBpedia Google Summer of Code 2013 project[23] for details.

The Wrapper Web Frontend

In Figure 5.4, the DBTropes wrapper frontend is shown. Note that due to space reasons, some information has been cut away—as shown in the resource summary at the top of the page, the page lists many more feature/trope instances and other information.

Page Controls

The page features caching controlling functions, features to control extraction hints (page type settings, hiding incorrect statements), and allows setting links

[22] In this context, a *category* is a link from the current Wiki page to another Wiki page that represents a category.

[23] http://blog.dbpedia.org/2013/11/29/making-sense-out-of-the-wikipedia-categories-gsoc2013/

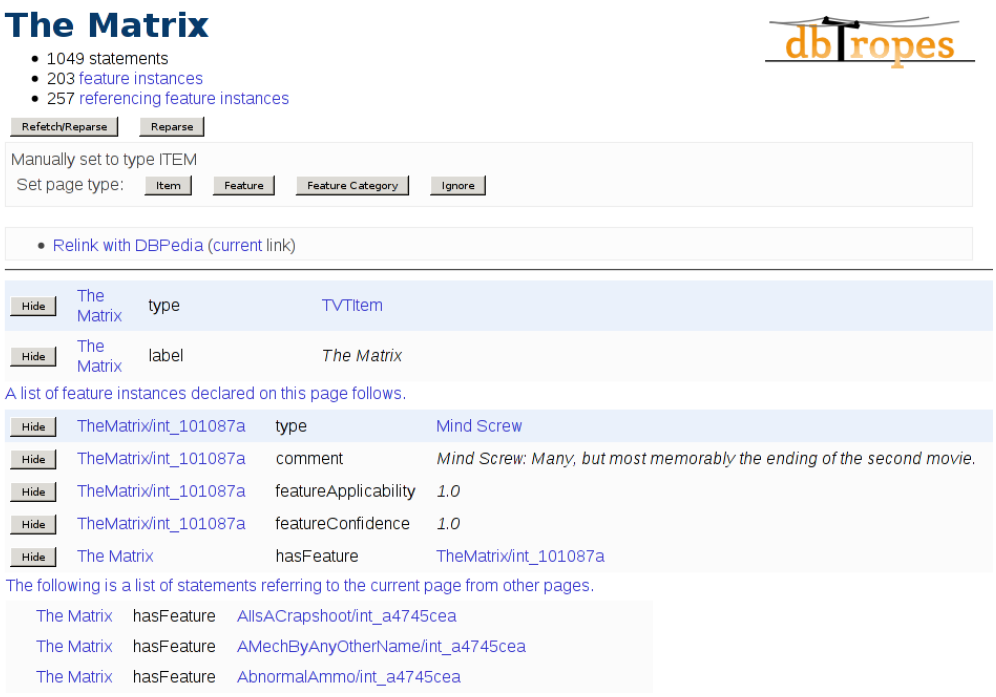


Figure 5.4.: (Shortened) screenshot of the DBTropes wrapper HTML frontend allowing immediate user feedback and fixes.

to other Linked Data Websites (DBpedia in this case—a link already has been set).

The actual extracted RDF data is shown in the form of (ordered) RDF triples. For DBTropes, this data is divided into three parts:

1. General information about the resource (type, label, comment, etc.).

2. Trope/Feature instance information.

3. Other resources linking to the current resource (i.e., statements about the current resource on other pages).

This presentation is not perfect for reading by humans. However, the frontend is merely intended to provide hints and let users correct errors. Applications that use the wrapper's data (e.g., Skipforward in the DBTropes case) typically use different domain-tailored renderers.

5.7.4. Consistency and Updates

To give an example for consistency problems in the TV Tropes/DBTropes scenario, the page type will be considered in the following. The page type determines how itemization lists in the HTML are interpreted: On item pages, links in itemizations are treated as potential feature instances, while on feature pages, links in itemizations are treated as potential item instances. In order to generate only RDF consistent with the Skipinions RDF schema, the extracted data is matched against this schema. "Incorrect" links (i.e., links to items in the itemization part of an item page) get dropped.

Concerning page updates, as detailed in Section 5.5, keeping cached wrapped data and the original data source in sync can be a challenge. Fortunately, TV Tropes exposes an RSS feed for page updates. DBTropes uses this as a basis to update its wrapped data.

5.7.5. Interlinking With DBpedia

Motivation

The ideas of the Web of Data is (i) making data available in machine-interpretable formats, and (ii) keeping links between data sources, enabling combining information from different data sources. As outlined in Section 5.7.1, TV Tropes features ample "soft" information about the items described, but most "hard" information such as the exact casts of movies or movie release dates is left away, as this information is readily available (to humans) via other sources such as Wikipedia or IMDb. In the Web of Data, the situation is similar: The "hard" data is available from sources such as DBpedia [LIJ+14] or Freebase[24] [BEP+08].

Challenges

Unfortunately, there is no straightforward way of finding out what TV Tropes articles correspond to what Wikipedia articles or IMDb entries: There are neither explicit links given in the HTML pages in general nor exist (reliable) inverse functional properties for the typical items of interest. Because of this, mapping approaches such as SILK [VBGK09] or the approach used for building LinkedMDB [HC09] do not work in this case. Instead, the method of choice is using a combination of generic approaches such as information extraction (parsing the article texts directly) and a number of heuristics.

In the following, the process used to relate TV Tropes entries with Wikipedia pages is described. The overall focus of the process was maximizing Precision; Recall is secondary only.

[24] `http://www.freebase.com/`

Overview of the Matching Process

Algorithm

For a given DBTropes resource, the first step is to identify candidate DBpedia resources that are potential matches. The alternative would be to compute pairwise similarity between all DBTropes resources and all DBpedia resources; the computation costs for these are prohibitive, and likeliness of false positives would be very high. Therefore, the label of the DBTropes resource is taken and used as input for Sindice [ODC+08] and DBpedia Lookup[25].This results in a first list of candidate matching resources. From this list, all resources that do not match the type of the DBTropes resource under consideration are removed: for example, movies in DBTropes cannot match books in DBpedia, and therefore any such potential matches get ignored a priori. Additionally, resources both on DBTropes and DBpedia that have several conflicting types get ignored. This is due to the fact that both sources occasionally mix resources of different types (e.g., a Wikipedia page that forms the basis of a DBpedia resource might describe both a movie and a book). Matching these resources is not desirable.

Similarity Metrics

From the remaining candidates, the candidate with the highest similarity to the DBTropes page under consideration must be chosen. Several similarity metrics get combined for this task:

- An Ontology-Based Information Extraction (OBIE) approach is used to extract semantics from the resource text descriptions.
- URI similarity metrics are used.
- Time information gets extracted and compared.
- Category information is mapped and used as evidence for similarity.

Then, the aggregated evidence is used to generate RDF statements that interlink the DBTropes resource with the candidate DBpedia resources. Only on strong evidence of a match, the *owl:sameAs* predicate is used for these interlinks, as false positives here have a lot of strong side effects. For weaker evidence, or if there is evidence that there *is* a relationship but it is not of the *sameAs* type but of another type (such as *part-of* or similar), other predicates are used.

Exploiting Transitivity

Finally, to get even stronger interlinking, also with different data sources, transitivity of the *owl:sameAs* relationship is exploited: If the DBpedia resource(s) found are, in turn, interlinked with a Freebase or a LinkedMDB resource, that *sameAs* relationship gets directly added to DBTropes.

In the following, the crucial steps in the process outlined above are discussed in more detail.

[25] `http://lookup.dbpedia.org/`

Handling of Types

False Positives

Handling of types/categories is crucial for preventing false positive matches. In the entertainment domain TV Tropes is mostly concerned with, many storylines are available both in movie and book (and possibly even board and computer game) form. A textual comparison of abstracts of these resources results in high similarity typically; even a semantic text content analysis results in high similarity as, after all, most content is shared between these distinct resources. However, a *sameAs* relation between a book and a movie, for example, is definitely to be avoided, as sameAs semantics would result in the two resources getting treated as one, resulting in a resource that is *both* a book and a movie—a resource that actually existed in neither of the both data sources originally.

Ultimately, very thorough checking of the types of the resources must be done. For implementing this comparison, a major challenge is lack and/or inconsistency of type information in the data sources.

- DBTropes resources typically have at least one category. Unfortunately, DBTropes categories are sometimes of limited use for type comparison (e.g., category The Epic).
- In DBpedia, resources may use several predicates to list information approximating categories (e.g., *dbp:type*, *rdf:type*, *skos:subject*). Ranges of these predicates differ.

Example

Consider the statement

dbtropes:Main/JerrySpringer *dbt:category1* dbtropes:Main/AmericanSeries

from DBTropes, stating that the Jerry Springer resource is of the American Series type, and the statement

dbpedia:The_Jerry_Springer_Show *rdf:type* dbpedia:TelevisionShow

from DBpedia, stating that the Jerry Springer Show is of the Television Show type.

Here, a human can easily deduce that both resources are essentially of the same type—both represent a television show. However, without further information, a machine cannot make this deduction as there is no apparent relation between the AmericanSeries type and the TelevisionShow type. While string comparison often helps with these problems, it is not necessarily accurate, and in cases just as the one presented it fails. Another problem is that type information can be semantically imprecise: for example, the DBTropes category AnimatedShows includes movies, TV shows, animes etc.; the category

Semantics of Type Information

FilmsOfThe2000s captures a whole decade. In DBpedia, some resources do not have any type information at all.

As mentioned above, both TV Tropes and Wikipedia pages often mix different actual resources; for example, they often describe both the movie and its associated book. If one of the data sources mixes several types but the other does not, an *owl:sameAs* relationship would be incorrect. However, other relationships *can* be used; for example, a *contains* relationship along with the (very generic) *rdfs:seeAlso* relationship would be appropriate.

Evidence for more specific kinds of relations can get extracted reliably by using type groups.

Type Groups

Looking at the example in the previous section again, one challenge is how to detect that both the AmericanSeries type associated using the *dbt:category1* predicate and the TelevisionShow type associated using the *rdf:type* predicate are actually representing related categories.

For reaching high Precision, lists of types that were found to represent related semantics were created manually. These lists are *type groups*. An example (shortened) type group for the TV show domain is as follows:

```
<CategoryGroup name="tv_show">
   <DbpediaTypes uri="http://dbpedia.org/property/type">
      <Type>tv</Type>
      <Type>tv series</Type>
   </DbpediaTypes>
   <DbpediaRdfTypes uri="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
      <Type>http://dbpedia.org/ontology/TelevisionEpisode</Type>
      <Type>http://dbpedia.org/ontology/TelevisionShow</Type>
   </DbpediaRdfTypes>
   <DbtropesCategories uri="http://dbtropes.org/ont/processingCategory1">
      <Type>http://dbtropes.org/resource/Main/AmericanSeries</Type>
      <Type>http://dbtropes.org/resource/Main/AnimatedShows</Type>
   </DbtropesCategories>
   <DbpediaSkosSubjects uri="http://www.w3.org/2004/02/skos/core#subject">
      <Type>http://dbpedia.org/resource/Category:Anime_of_2008</Type>
      <Type>
         http://dbpedia.org/resource/Category:Animated_series_based_on_video_games
      </Type>
   </DbpediaSkosSubjects>
</CategoryGroup>
```

Building Type Groups

Unfortunately, assembling type groups manually is a tedious process. While this can be automated theoretically (essentially, this is a subfield of the *Ontology Mapping* domain), for the problem at hand too little meta information is present—any algorithm finding matches in types has little more to work with than the bare URIs and their labels, and instances for these types. Typical ontology mapping algorithms would resort to string comparison for type labels, and instance comparison for instances of the types—which leads us to a chicken and egg problem in this case.

In the end, a type group file of moderate size was built manually. The complete XML file used by the matching algorithm contains 18 type groups and 225 different types. During testing it became clear that the more entries the type groups contain, the more precise the algorithm works, and even the small type grouping file allowed to increase matching accuracy considerably.

Cardinality

Since the primary goal is to prevent false positives and, thus, filtering out resources with incompatible types, types do not need to be unique in the list of groups: One type may appear in more than one type group: for example, the DBTropes category AnimatedShows may be used for films *and* television shows. Therefore, the AnimatedShows category was added to both the `film` and the `tv show` type group. If a page has the AnimatedShows type, it gets assigned to both `film` and `tv show` type groups. If a page has multiple types, it will be associated with all the corresponding type groups.

As previously mentioned, the algorithm is able to not only create *owl:sameAs* relationships but also other relations. To explain this in a sound way, the notion of contradicting types and type compatibility needs to be introduced formally.

Contradicting Types

Different Domains

A resource has *contradicting types* if two or more types of that resource are associated with type groups of different domains. Since every type can belong to several type groups, type compatibility checks must be done for all of these groups: for example, the type AnimatedShows belongs to the `film` and `tv show` type groups. Any other type must belong to `film`, `tv show`, or both type groups: The set of type groups for each type must be a subset of the union of all type groups.

The function `map` maps types to a set of type groups. In the above case, `map(AnimatedShows)` would result in `[film, tv_show]`.

Let n be the number of types of a resource and t_n the nth type of the resource. Then the following Boolean function checks if the types $t_1, \ldots, t_n$ are contradicting.

$f(t_1, \ldots, t_n) = \bigvee_{j=1}^{n}((\bigcup_{i=1}^{n} map(t_i)) \supseteq map(t_j))$

The same expression, reformulated for easier implementation:

$f(t_1, \ldots, t_n) = |\bigcup_{i=1}^{n} map(t_i)| > max_{i=1}^{n}|map(t_i)|$

The following examples illustrate how the function works:

$A =$ AnimatedShows type
$F =$ Film type

$M =$ Manga type
$G_F =$ `film` type group
$G_{TV} =$ `tv show` type group
$G_L =$ `literature` type group

The resource has types AnimatedShows and Film:

$f(A, F) = |map(A) \cup map(F)| > max(|map(A)|, |map(F)|)$
$f(A, F) = |[G_F, G_{TV}] \cup [G_F]| > max(|[G_F, G_{TV}]|, |[G_F]|)$
$f(A, F) = |[G_F, G_{TV}]| > max(2, 1)$
$f(A, F) = 2 > 2$
$f(A, F) =$ false (not contradicting)

The resource has types AnimatedShows and Literature:

$f(A, M) = |map(A) \cup map(M)| > max(|map(A)|, |map(M)|)$
$f(A, M) = |[G_F, G_{TV}] \cup [G_M]| > max(|[G_F, G_{TV}]|, |[G_M]|)$
$f(A, M) = |[G_F, G_{TV}, G_M]| > max(2, 1)$
$f(A, M) = 3 > 2$
$f(A, M) =$ true (contradicting)

Meanings of Contradicting Types

Depending of the combination of contradicting types detected, different outcomes are possible.

- The DBTropes resource has contradicting types: The DBTropes resource likely actually is a conglomerate of several resources (book, film, game, ...). It is very unlikely an exact matching DBpedia resource exists. This resource gets excluded as a candidate for exact matches.

- The DBTropes resource has contradicting types; a candidate DBpedia match exists that matches a subset of the DBTropes type (groups). The DBpedia resource is possibly contained in the DBTropes resource.

- Vice versa: The DBpedia match exhibits contradicting type groups, and the DBTropes resource matches a subset of these: The DBTropes resource is possibly contained in the DBpedia resource.

Since the focus is on exact matches, type *compatibility* (not subsumption, etc.) is important. The next section focuses on this.

Compatibility of Types

A resource has *contradicting* type groups if these span across multiple domains. Contradicting types are a strong indication that (if present for one resource) the resource is actually a composite resource, or (if the type groups of one resource of DBTropes and one resource of DBpedia are contradicting) the resources are no match. However, if types are not contradicting, this is no indication for a match: Non-contradicting types are a *necessary* prerequisite for matching (as in *owl:sameAs*) resources, but non-contradicting types are no *sufficient* criterion.

Composite Resources

Another necessary prerequisite for a match is *type compatibility*. To check type compatibility, the set of type groups for the DBTropes and DBpedia resources in question are looked up. The resources have compatible types if the set of type groups of one page is the subset of another.

Let n be the number of types in a page, t_n and d_n the nth type in the DBTropes and DBpedia resources, respectively. The following Boolean function checks if the resources are type compatible:

$$f(t_1, \ldots, t_n, d_1, \ldots, d_n) = (\bigcup_{i=1}^{n} map(t_i) \subset \bigcup_{i=1}^{n} map(d_i)) \\ \vee (\bigcup_{i=1}^{n} map(d_i) \subset \bigcup_{i=1}^{n} map(t_i))$$

One corollary is that resources that have no types associated are compatible with all other resources. In this approach, passing this step is necessary for generation of *sameAs* relations; *contains* and *part-of* relations are not restricted by this step.

For example, consider one page belonging to the `film` type group and the other page belonging to both `film` and `literature` type groups. In this case the types of the pages are compatible and the contradiction of types is used to decide the kind of relation (e.g., *owl:sameAs*, *contains*).

Example

Other Similarity Metrics

The types of the resources in question represent a strong source of evidence for elimination of possible match candidates, and for subsumption and other relations. However, type compatibility alone is no strong evidence for an actual semantic match, even when considering the original text lookup that lead to the potential match resources.

The procedures explained in the previous paragraphs yielded a list of DBpedia match candidates for the DBTropes resource in question. These candidates need to get filtered more to find the best candidate for a match. Three similarity metrics have been implemented to achieve this goal:

- Ontological text comparison based on ontology-based information extraction. This compares text abstracts (or, to be more specific, the *rdfs:comment* statements generated from the text abstracts available from both Wikipedia and TV Tropes) in a way that adjusts for the big differences in wording prevalent in Wikipedia and TV Tropes.

- A URI similarity metric. Both Wikipedia and TV Tropes encode additional information in the URIs of their articles such as type hints and release date/year. The metric takes this into account.

- A release date similarity metric. Since in the entertainment domain many series and remakes exist, accuracy concerning the time of creation of resources is important. In DBpedia/Wikipedia the release date is typically modeled well, but in TV Tropes/DBTropes, these dates are often available in non-canonical form only.

Text Comparison Using Ontology-Based Information Extraction

Both DBpedia and DBTropes include the text abstracts of the wrapped information resources/Wiki pages as *rdfs:comment* statements. This text is the main part of the page, and typically the most important part for human readers. Text comparison of these texts seems a reasonable approach to deduce resource similarity. However, a major challenge here is that the writing style of Wikipedia and TV Tropes is very dissimilar. See Figure 5.5 for a TV Tropes movie article[26].

Writing Style

Note the writing style is very informal; many TV Tropes articles contain long passages of insider jokes and background information, whereas Wikipedia articles typically strictly list the cast and a plot outline. Standard string comparison metrics (e.g., N-Gram, Levenshtein distance, TF/IDF) struggle with this kind of problem, as it makes texts from TV Tropes and texts from Wikipedia very dissimilar.

As a way to counter this, instead of string comparison, ontology-based information extraction was used.

> "Ontology-based information extraction (OBIE) aims at extracting formal facts from natural language texts. In terms of RDF, facts may be triples representing attribute knowledge about a resource or relations between resources." [RBAD10]

[26] `http://tvtropes.org/pmwiki/pmwiki.php/Film/TuckerAndDaleVsEvil` – accessed November 2013

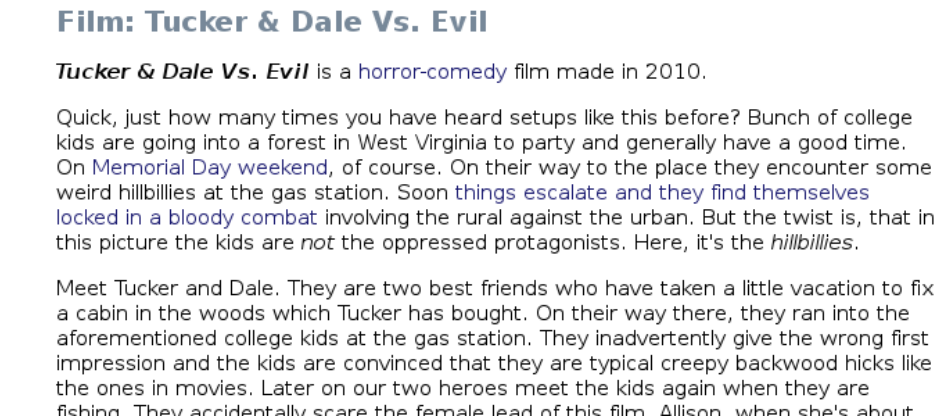


Figure 5.5.: A TV Tropes movie article. Notice the informal writing style.

As OBIE implementation, SCOOBIE (ServiCe OntOlogy-Based Information Extraction) [vEDAH09] was employed. SCOOBIE is an ontology-based information extraction system that uses symbolic background knowledge for extracting information from text.

OBIE Background Knowledge

SCOOBIE extracts information from texts, based on a background ontology (DBpedia in this case). That is, given a text and a background ontology, SCOOBIE returns lists of ontology instances it found evidence for in the text, including belief values representing how certain SCOOBIE is about the occurrence of the respective instance. SCOOBIE uses linguistic rules (by means of regular expressions) and gazetteer lists (word lists that allow to perform named entity recognition) to extract information from given texts [Ebe10].

SCOOBIE was used to extract instances from both comments and labels of DBTropes and DBpedia resources. Then, a similarity value was calculated based on the number of shared instance occurrences in DBTropes and DBpedia, divided by the number of instances found in the DBTropes resource comment.

General Characteristics

In the end, this approach heavily weights individual keywords in the texts compared, and thus is very different from normal string comparison. By design, it ignores stopwords and any text not occurring in the background ontology. For the use case at hand, this is appropriate: Any occurrences of actors will be spotted (as actors are modeled in the background ontology DBpedia); the same is true for directors, studios, geographical locations, names of holidays, etc., while potentially misleading information such as trope names get ignored.

URI Similarity Metrics

To further increase accuracy, another metric was implemented. The URI similarity metric compares the last segment of the URIs in question, as correct matches typically have similar to almost identical URIs in this regard. The last segments of the URIs are converted to lowercase alphanumerical strings and compared using the 4-gram string comparison metric. DBpedia pages often include the resource type in their URIs. To account for this and to improve accuracy, the category of the DBTropes resource is considered in the comparison as well.

Example

Consider this example:

- DBTropes resource: dbtropes:Animated/TheHobbit
- DBTropes category: dbtropes:FilmsOfThe1970s
- DBpedia resource: dbpedia:The_Hobbit_(1977_film)

Here, the strings "thehobbit"+"filmsofthe1970s" get compared with "thehobbit1977film". The N-gram metric works by generating all four character substrings for the compared strings, then counting the substrings that occur in both the compared strings. The metric is calculated straight-forward, by counting the 4-letter substrings of the DBpedia URI which are either contained in the DBTropes URI or one of the DBTropes category strings. In this example, the result is 7: The matching substrings are "theh", "heho", "ehob", "hobb", "obbi", "bbit", and "film".

Release Date Comparison

A special challenge in the entertainment domain concerning finding semantically equal resources is that due to remakes (for films) or changing publishers (for comics), many resources exist that look very similar if comparing their normal properties. By "normal properties", here content/plot keywords, character names, and actors are referred to. However, it must be noted that in TV Tropes, actors are often not mentioned, leaving the algorithm with only plot-related properties to work with, which makes remakes look very similar to each other. The one discerning property here is *release date* which, if different for two resources, is a strong indication that those resources do not match. While DBpedia typically models the release date reasonably well, this is typically not the case in TV Tropes/DBTropes. The problem there is that TV Tropes does not use a canonical format for release dates; often, for example, the release date is hidden

Differences in Modeling

in a category the resource in question is associated with. The label of such a category can take the form of FilmsOfThe1970s or even the form of TurnOfTheMillenium. So, not only are release dates not easily available in TV Tropes, they are often ambiguous since they are only narrowed down to a time interval. The similarity metric has to take this into account.

Comparison Metric

The year comparison metric, as implemented, is based on a variant of the *Earth Mover's Distance measure*[27]. An integer array ranging from 1900 to 2099, each representing a potential release year, is created for both the DBTropes resource and the DBTropes candidates. For any evidence of a release date found in the resources, the corresponding field of the array gets incremented by a weighting value.

Some DBpedia pages contain a predicate *dbp-ont:releaseDate* expressing a high-confidence release date. To reflect this, the field in the DBpedia array representing that year gets incremented by a high value.

DBTropes resources do not feature high confidence release date properties. However, release year candidates can be extracted from DBTropes categories such as TheNineties or FilmsOfThe1970s. A number of heuristics has been built that allows adding the release date evidence to the year integer array: for example, if the string "1970s" is found, the array fields representing the years 1970 to 1979 get incremented.

Array Comparison

At the end of the calculations, two integer arrays with 200 integer values each have to get compared. The final similarity is calculated by going through the 200 pairs of array fields and calculating the total sum of the minimum value of each of these pairs.

Creation of Relations

After calculating the similarity metrics for all DBpedia candidates, the candidate with the highest similarity value is marked as best match. The actual type of relation to the DBTropes resource is determined depending on several criteria. Four general types of relations can be created depending on the result of the calculation of the contradiction of types: same-as, contains, is-part-of, and related. If applicable, same-as is immediately transformed into an *owl:sameAs* property; the others are persisted in own namespaces. The following table shows how the relation type gets determined.

[27] `https://en.wikipedia.org/wiki/Earth_mover's_distance` – accessed November 2013

Contradicting DBTropes Types	Contradicting DBpedia Types	Statement
No	No	same-as
No	Yes	is-part-of
Yes	No	contains
Yes	Yes	related

A number of additional criteria are employed to further decrease the number of false positives.

- A threshold value must be exceeded for a match to be considered a *owl:sameAs* candidate. This threshold value was experimentally determined using a gold standard. If the threshold is not exceeded, no *sameAs* link is considered.
- If the second best match has a similar match value, it is likely that the algorithm would choose a random resource out of a number of very similar resources for *sameAs* linking. In the light of the problems with series and remakes, this is not desirable. Therefore, a relative threshold between the best match and the second match must be met.

The relative threshold is calculated by dividing the similarity value of the best match by the similarity value of the second match. If the result is above the threshold, an RDF statement with the best match according to the above table is created. Otherwise, predicates that reflect this lack of certainty are created for all the top matches falling in the relative threshold range of the top match. The following table summarizes the behavior:

Best match above abs. threshold	Best match above rel. threshold	Contradicting DBTropes Types	Contradicting DBpedia Types	Statement
Yes	Yes	No	No	same-as
Yes	Yes	No	Yes	is-part-of
Yes	Yes	Yes	No	contains
Yes	Yes	Yes	Yes	related
Yes	No	No	No	possibly-same-as
Yes	No	No	Yes	possibly-part-of
Yes	No	Yes	No	possibly-contains
Yes	No	Yes	Yes	related
No	-	-	-	-

This concludes the presentation of the DBTropes/DBpedia instance matching algorithm.

5.7.6. Evaluation

Precision in Facts Extraction

Trope Extraction Process

For evaluating Precision and Recall of the trope extraction process, a manual check of the data extracted was carried out. This covered randomly selected item and trope pages with about 580 trope occurrences all in all.

In the test set's *trope pages*, 460 trope occurrences were counted manually. About 100 of these were deemed not to be extractable automatically in any case (because the items mentioned were present only as plain text and not represented as a Wiki link, etc.). DBTropes extracted 300 trope occurrences. Most trope occurrences identified but not extracted (58) were due to DBTropes not having enough data to estimate the type of the page linked to—in this case, DBTropes errs on the side of caution and drops the statement, giving a notice. This leads to a Recall of about 83%. Of the extracted occurrences, 14 were invalid, yielding 95.3% Precision.

In the test set's *item pages*, 120 trope occurrences were counted manually. Apart from 4 of them, all seemed extractable with reasonable effort. DBTropes extracted 104 trope occurrences, having dropped 11 occurrences due to missing type data. Four of the extracted occurrences were invalid. This leads to 89.7% Recall and 96.2% Precision. In typical Information Extraction tasks such as those run in the context of the DBpedia project, Precision values exceeding 90% are typically deemed acceptable.

This shows that rigid application of domain rules leads to high Precision in facts extraction. **Hypothesis H4 holds.**

All of the invalid occurrences could be removed using the interactive DBTropes features after measurements.

Conclusion

The conclusion of this is that item pages are easier to parse than trope pages. This is not unexpected: When adding items as occurrences to a trope page, people are likely to add whatever movies and other items they can think of, which is an almost unlimited set. In contrast, when adding tropes to item pages, people are likely to use only tropes defined in TV Tropes.

The quite good Recall and Precision results of DBTropes are also thanks to the very rigid style the TV Tropes community employs for collecting data. Despite TV Tropes being a collaborative fun project, the overall style of the Wiki is very consistent, and domain knowledge of the people involved is very high. In the TV Tropes forums, a group of specialized people work on the structure

of TV Tropes, collaboratively doing effectively *Ontology Engineering* as well as *Ontology Evolution*, which contributes to the high quality of the data available.

Interlinking With DBpedia

Gold Standard

The algorithm interlinking the DBTropes data source with DBpedia (Section 5.7.5) has been built and optimized based on a **Gold Standard** sameAs mapping of 150 resources of DBTropes to 141 resources in DBpedia[28]. This gold standard has been built manually. As outlined in the motivation, the main goal was to build an algorithm that heavily focuses on Precision and sacrifices Recall if necessary. Several alternative ways for implementing each of the steps outlined in the chapter have been tested: for example, for string comparison the *Text::Similarity* Perl module[29] has been considered. From its homepage:

> "This is a Perl module that measures the similarity of two files or two strings based on the number of overlapping (shared) words, scaled by the lengths of the files. It computes the F-Measure, the Dice Coefficient, the Cosine, and the Lesk measure."

However, it was found that for the purpose of comparing Wikipedia abstracts with TV Tropes abstracts, it performed worse than the OBIE-based approach.

Target Precision

As target Precision during developing and tuning the approach, the number of correctly found *sameAs* matches was supposed to meet or exceed 90%, measured on the gold standard. This Precision value was chosen as a compromise allowing to keep development and tuning effort manageable, at the same time providing results that serve as a useful base for possible later refinement using manual intervention in the DBTropes frontend.

During tuning of the approach, tweaking URI distance metrics as outlined previously yielded a 12.5% improvement of Precision; later refinements of type categories, and adding year comparison metrics resulted in overall Precision exceeding the 90% goal.

Persisting Results

Once the goal was met, the calculated match relations were persisted in DBTropes, along with *sameAs* relationships fetched from sameas.org. This resulted in 6,000 *owl:sameAs* relations between DBTropes and DBpedia (November 2010); at the time, DBTropes featured about 16,000 items. More than 800 *owl:sameAs* relations to Freebase.org were fetched from sameas.org; for LinkedMDB, more than 450 *owl:sameAs* links were generated.

[28] The missing nine resources are due to non-existing match resources in DBpedia.

[29] `http://www.d.umn.edu/~tpederse/text-similarity.html` – accessed November 2013

5.8. Conclusion

This chapter gave an overview of several approaches providing Linked Data from existing Websites. Special consideration was given to the online wrapper approach, detailing its application for the TV Tropes Website. The resulting DBTropes.org wrapper is a prototype implementation that has been created to demonstrate the features of the online wrapper approach. It successfully demonstrates large-scale semantic data wrapping of an existing (but non-semantic) data source. DBTropes gets updated continuously and provides high quality data with a focus on high Precision. In contrast to other similar approaches, it operates in real time, and allows incorporating user feedback to improve the facts extraction process.

Online Wrapper

Another key characteristic of DBTropes is that it uses Skipforward's *Skipinions* ontology for modeling the data retrieved from TV Tropes. Consequently, DBTropes can serve as a source of information for Skipforward.

For further enriching the data made available by DBTropes, an approach for interlinking DBTropes resources with DBpedia resources has been presented. Interlinking both data sources allows data consumers to run expressive queries against the combined data of both sources. Since the focus of both data sources is different yet the resources described in both sources are overlapping, the combined expressivity is very high. The interlinking approach, just as the actual data extraction approach, strongly favors Precision over Recall to keep noise minimal. The approach differs from existing tools intended for the same purpose: It uses custom-tailored text comparison based on ontology-based information extraction as well as several specific metrics that handle challenges such as fuzzy date specifications. The approach was developed and continuously evaluated using a manually built gold standard. It reaches the target expectations concerning Precision as required.

Interlinking With DBpedia

IV

Evaluation

CHAPTER 6
Evaluation

6.1. Goals

The purpose of this chapter is to investigate the validity of the hypotheses presented in Section 1.2. Where possible, the evaluation seeks to quantify any findings. For convenience, here the hypotheses are listed again.

- Hypothesis H1: The base ontology, implementing a feature hierarchy and supporting truth values and user confidence, allows intuitive expressive annotation of resources in a multi-user scenario.
- Hypothesis H2: Support when creating ontology-based annotations leads to higher annotation volume and, in general, better user acceptance.
- Hypothesis H3: Embracing inconsistencies and providing personalized views leads to improved user experience and better item recommendations.
- Hypothesis H4: Validation using domain rules results in high Precision in ontology and facts extraction.

Note that H4 was covered in Section 5.7.6 already. Only H1 to H3 are covered in the current chapter.

6.2. Challenges

The document-based scenario mostly employs ideas and functionality that are enabling technologies. Quantitative analysis and evaluation, thus, is difficult; qualitative user feedback is more valuable and meaningful than comparing statistics from wholly different (non-semantic vs. semantic) systems.

The following evaluation approaches are especially noteworthy (see [QHK03] for a discussion of evaluation approaches in similar domains):

- **Controlled experiments** in which test users are given tasks they are to complete under supervision. Since with complex functionality there is a learning curve, the approach is less suited for evaluating complex functionalities. However, for testing specific limited functionality, this approach is appropriate. Since Kaukolu has been part of several projects in which related functionality has been evaluated using controlled experiments already (see below), no further experiments were part of this evaluation.

- **Long-term usage and user interviews** allow iterative improvement of the software and collecting precise feedback even for complex functionality. In specific project contexts, Kaukolu has been subject to this approach. For the hypotheses relevant in this thesis, no additional long-term testing was necessary. User interviews, however, were carried out, partially focusing on the Wiki functionality, and immediate user feedback during the evaluation led to software improvements.

Kaukolu Evaluation

See [Sau09] for an evaluation of Kaukolu in the context of the Semantic Desktop/PIMO; the reading/attention detection functionality of Kaukolu is equivalent to the one presented and evaluated in [Bus10]; user context elicitation has been presented and evaluated in [Sch10]. The annotation recommender functionality that comes into play in the evaluation setup chosen here is based on Skipforward's annotation recommendations which have been evaluated in detail in [Ami12]. Therefore, the evaluation of the document-based scenario in this chapter focuses on qualitative feedback gathered in the final user interviews.

Evaluating recommender engines is a difficult challenge. There is plenty of literature in the area of recommender engine evaluation [HKTR04, SG11, STL11, HdOG08, Pow07, SK11]. In case there are simple data models and clearly defined goals of the recommender—such as in collaborative filtering scenarios featuring only user liking per item—evaluation is straightforward. Given a ground truth dataset is available, the following approaches can be pursued:

Ground Truth

- For predicted liking values, root-mean-square error metrics can be calculated. This has been done in the Netflix prize[1], for example.

- For strictly positive/negative recommendations, one can calculate Precision, Recall, and F-Measures on the resulting data.

[1] `http://www.netflixprize.com/`

- For the same preliminaries, calculating a Receiver Operating Characteristic curve (TPR against FPR for all set sizes of recommendations) and the associated Area Under Curve (AUC) is possible.

Skipforward Scenario

Unfortunately, several factors make these approaches inappropriate for evaluating the Skipforward recommenders:

- For the small dataset and the complex recommendation profiles supported by the Skipforward approach, a strict distinction between a "true" and a "false" recommendation is not suitable as this would represent an arbitrary cutoff for the ordered list of recommended items.

- The content-based/hybrid recommender nature of Skipforward and its resulting flexibility concerning the actual recommendation goal makes building a generic ground truth infeasible. Limiting the evaluation to item liking only would ignore most of the potential of the approach.

Cross-Validation

It has to be stressed that the actual item recommender algorithm is not of interest in this thesis. In fact, Skipforward uses a rather simplistic algorithm for item recommendation. Hypothesis H3 does *not* make any claims concerning a specific recommender algorithm though; it merely claims that usage of *subjectively weighted* annotations as *input* for an item recommender increases its *result* quality. Consequently, if it can be shown that Skipforward's weighted approach achieves higher data quality for the recommender input than a non-weighted approach does, it follows that recommender output has higher quality as well, proving the validity of hypothesis H3. Therefore, instead of evaluating recommendations, it is possible to **evaluate recommender input annotation quality** to verify hypothesis H3. This can be done using a **cross-validation** approach by ignoring part of the user annotations while generating Skipforward's personalized views, treating the ignored data as ground truth, measuring the error, and repeating these steps for a non-weighted approach.

Evaluating user experience is typically done using approaches such as the Technology Acceptance Model (TAM) [Dav85] and psychometric methods [Lew95].

TAM and Related Models

The basic TAM (see Figure 6.1) builds on the assumption that the user's *attitude towards using* a system is a major factor in the actual system use. The attitude towards using a system, in turn, is expected to be influenced by the *perceived usefulness* of the system and the *perceived ease of use*. These two beliefs are supposed to be directly influenced by system design characteristics.

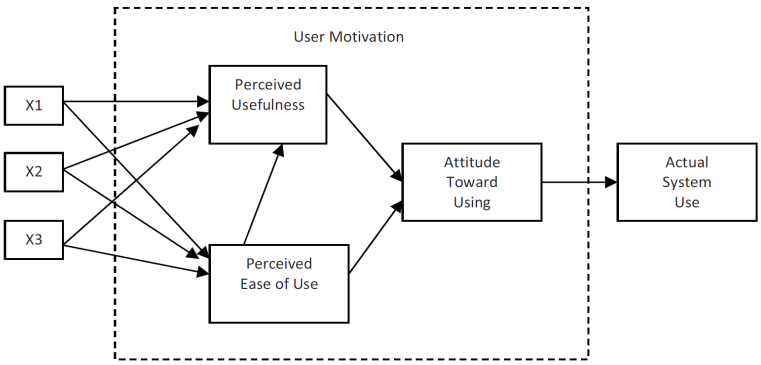


Figure 6.1.: Original Technology Acceptance Model. [Dav85]

A number of models influenced TAM or build upon it: The Theory of Reasoned Action (TRA) [FA75] features a *Subjective Norm* factor that is mostly influenced by *Normative Beliefs and Motivation to Comply*, representing external factors; a revised version of TAM in 1993 [Dav93] found additional links between factors such as system design characteristics directly influencing the user attitude towards using the system; the Theory of Planned Behaviour (TPB) [Ajz75] investigates *Control Beliefs and Perceived Facilitation* as a contributing factor concerning user behaviour. The Unified Theory of Acceptance and Use of Technology (UTAUT) [VMDD03] investigates shortcomings of existing models and proposes a unified theory outperforming the previous models in an evaluation. To give an example, for measuring the *Attitude Toward Behaviour* construct, statements like "Using the system is a good idea." and "Using the system is pleasant" are used.

Psychometric Models

In comparison, IBM's Post-Study System Usability Questionnaire [Lew95] features questions such as "Overall, I am satisfied with how easy it is to use this system.", or "I believe I could become productive quickly using this system.".

TAM Aims

These questions show that the models mostly target evaluating overall user acceptance of a system. No specific aspects of the system are investigated: The design and usability of the user interface influences the model output, as (in some models) do aspects such as peer pressure. Additionally, typically these approaches are used when introducing entirely new systems (e.g., Davis used TAM to evaluate user acceptance of electronic mail in 1989).

Goals of the Skipforward Evaluation

In the evaluation of Skipforward, the focus lies on individual aspects of the system. While many parts of the Skipforward approach are novel, the generic idea of having a resource-centric information repository is not; therefore, asking users whether they like to use Skipforward in general is likely to give less than

helpful results (for example, comments would likely include "No, I will rather use Amazon, because there I can immediately buy items of interest."). Consequently, the evaluation has to be limited to specific aspects such as whether the Skipinions data model is suitable for its task, and whether the system's handling of trust is perceived as useful by the user. Specifically, evaluation of the user interface was limited to the parts integral to the Skipinions data model (i.e., applicability and confidence handling). Since Skipforward is intended as a research prototype for small user groups consisting of domain experts (who accept and work around user interface quirks to some degree), the focus when developing Skipforward was developing functionality rather than investing a lot of time in refinement of the user interface.

Resulting Evaluation Design

In the end, the evaluation of Skipforward was carried out using a mix of quantitative and qualitative measurements. Rather than measuring overall user acceptance, questions similar to those in TAM and related models were used to find out user acceptance (or, in the wording of TAM, *perceived usefulness*) of specific features of the Skipforward system. I believe this maximizes insight concerning Skipforward's novel features.

6.3. Evaluation Setup

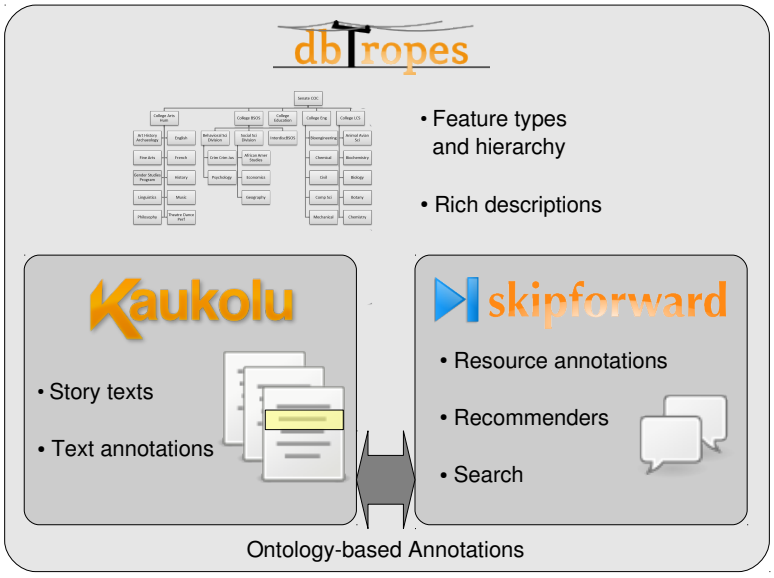


Figure 6.2.: Evaluation setup.

In the 2013 evaluation, as depicted in Figure 6.2, a combined system of Kaukolu, Skipforward, and DBTropes was used. Users were given the task of reading a set of SciFi/Fantasy short stories in the Wiki, annotate the stories with feature types and tropes both in the Wiki and in Skipforward, and to test search and recommender functionality. In this setting, the individual systems were used as following:

- Kaukolu was used as document storage and document annotation system.
- Skipforward served as annotation recommender and item and expert recommender as well as frontend for semantic search.
- DBTropes served as a source of feature types.

Kaukolu was extended with functionality to **create Skipforward annotations**: Users were able to create new text annotations with annotation types fetched from the Skipforward system on-the-fly. Annotations then got stored both in Kaukolu and in Skipforward. This allowed the user to express facts such as "In *this* paragraph, an Action Girl occurs" or "*This* paragraph contains a lot of Humor". At the same time, features of both systems can be used: In Kaukolu, the user was able to use advanced search to find all text passages annotated with Action Girl; Skipforward supplied annotation recommendations (also passed through to Kaukolu) and item recommendations ("If you liked *this* story, you should read *that* story as well").

Datasets

Concerning datasets, Kaukolu contained **25 short stories** (plus some extra stories for people who wanted to continue reading). These stories were mostly imported from the Interzone SciFi/Fantasy short story magazine[2] (issues 241 to 246). Some very long stories have been excluded to keep reading times reasonable; the set of core stories has an average story length of 5,000 words which results in about 15 minutes reading time per story. Additional to the Interzone short stories, a few other short stories of the same domain have been added. The list of the 25 core short stories can be found in Appendix C.

Ontologies

Concerning **ontologies and annotations**, Skipforward contained a subset of the DBTropes trope hierarchy[3] and the *Storyteller* domain ontology, containing some basic and explicitly subjective feature types. These **31 subjective feature types** included types such as *Convincing Plot*, *Good Twists*, and *Intriguing Writing Style*. A complete list of these feature types is available in Appendix C. The short stories present as Wiki pages in Kaukolu had not been annotated in Skipforward (yet).

Participants

Concerning the **users** participating in the evaluation, the idea was to create

[2] `http://ttapress.com/interzone/`

[3] The hierarchy was built by the DBTropes Skipforward synchronization bot, see Section 5.7.3.

a community of long-term Interzone readers and collect statistics and feedback from them initially. To these ends, I contacted TTA Press in 2011, and thankfully got support from Roy Gray who put a Skipforward advertisement in several issues of Interzone eBook releases. Additionally, I advertised Skipforward in the TTA Press online Web forums[4]. Unfortunately, feedback was limited, and no long-term community could be built in the end. That might be partially due to the prototype character of Skipforward, but I suspect that the outreach of the advertisements and forum threads has been limited as well: The TTA Press forums have low traffic in general, and the main release channel of Interzone is the printed edition; advertisements were present in the eBook variants only though.

In the end, I contacted a number of people I thought might be interested in participating in the evaluation. This included co-workers from the DFKI Knowledge Management department who have a knowledge worker background, but people with other backgrounds participated as well (e.g., a librarian, a journalist, a mathematician). **15 people** filled out the initial questionnaire; **ten people** completed the evaluation. Of the ten people completing the evaluation, five had a knowledge worker background.

Reading approximately 130,000 words (about seven hours reading time) and annotating the stories was quite a tall order. I would like to thank all participants. To provide a bit of external motivation, prizes (see Figure 6.3) were given out for reading and annotating the 25 core stories. The prizes consisted, among other things, of about 4kg of high-quality chocolate and 1.3kg of other sweets. All in all, they have an estimated calory count of 34,000kcal (not including beverages).

6.4. Evaluation Process

The **evaluation process** included several steps carried out by each evaluation participant. The exact evaluation participant instructions can be found in Appendix B.

Filling out a questionnaire that was primarily focused on finding out the user satisfaction concerning the state of the art in recommender systems, goal-directed search, and related annotation-driven functionality. This was done to verify and quantify the need for advanced annotation/trust-based services, and

[4] `http://ttapress.com/forum/`

Figure 6.3.: Evaluation prizes.
"Motivation *chapter*?- I have a motivation *picture*!"

to find out if there is any further functionality that might be worth pursuing in the evaluation. Users were asked to quantify their satisfaction with and the relevance of these features in existing systems such as Amazon. The questionnaire can be found in Appendix A.

Reading and annotating stories in the Wiki. In the context of the evaluation, reading all stories online in the Wiki would have been the preferred option. However, as this is time consuming and requires the participants to spend a lot of reading time in front of the computer, reading printed versions (partially) was allowed as well.

Annotation can be carried out by various means and was typically a process: In the Wiki, Skipforward annotations can get created directly. However, one can also start by just adding free text annotations (present in Kaukolu only and not duplicated in Skipforward). In a later step, the user can create Skipforward annotations that represent the original annotation. This is handy if the user wants to keep reading but mark a passage as potentially "trope worthy".

Continue annotating in Skipforward. After creating annotations from within the Wiki, users were asked to switch to the Skipforward system and have a look at the annotation recommender functionality there. In the process, Skipforward

programmatically made sure that the subjective feature types present in the Storyteller annotation ontology were assigned by each participant.

Skipforward search for any combination of feature types. This gives the user a better idea of what is possible using the Skipforward data model.

User interviews and questionnaires collected final user feedback concerning the perception of the system, the estimated potential of the new approaches and functionalities, possible improvements, as well as problems encountered during the evaluation.

Overview of Collected Data

A lot of different data has been collected during the evaluation. **All data is openly available** on skipforward.net, using Linked Data APIs. Additionally, the installation features a demo account that allows browsing the data using a normal Web browser, and without requiring to create a personalized account.

User Questionnaires and Interviews — At several points in the evaluation, participants were asked to fill out questionnaires, and interviews were carried out. It has to be noted that in the result presentation, feedback concerning the user interface was kept to a minimum, as due to the prototype nature of the system UI design was of secondary importance.

User feedback gave indications on evaluating hypotheses H1 and H2.

Wiki Annotations — In the Wiki, several types of text annotations were available. (i) *Text annotations* or *Tags* for quick-and-dirty annotations that later potentially may get refined to... (ii) *Skipforward annotations* that use Skipforward's feature types. This was implemented by creating a normal ontology-based Wiki annotation, based on a Skipforward annotation class, and creating an annotation in Skipforward via a Web REST call in parallel. Annotations created this way include a backlink from Skipforward to the Wiki. Only the ontology-based Skipforward annotations were available for later recommendation tasks.

In Figure 6.4, a Wiki page containing one story is shown as it looked like after the evaluation was over. Several users read the story and created annotations in it. The annotations highlighted in yellow are two overlapping annotations, both created by evaluation participants, using Skipforward feature types.

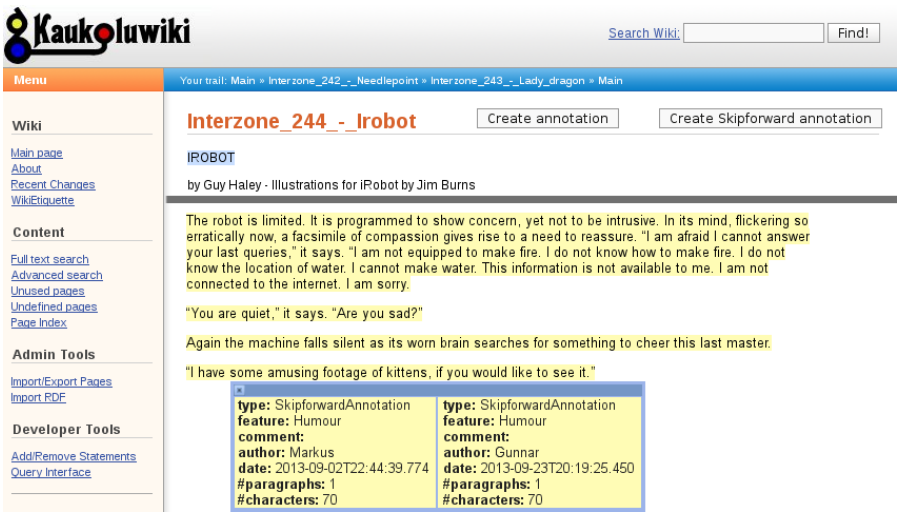


Figure 6.4.: Annotations created in Kaukolu.

Skipforward Annotations — *Skipforward annotations* could get created in the Wiki and got duplicated automatically in the Skipforward system. Annotations created directly in Skipforward were only available in the Skipforward system. There are multiple ways of creating annotations in Skipforward; any annotation created carries provenance information that denotes how the annotation was created.

Later analysis of Skipforward annotations allows validating hypothesis H3.

Provenance Information — For all annotations created and for some user interaction, provenance information was tracked.

- Who created the annotation?
- When was the annotation created?
- In what context was the annotation created? (during what phase of the evaluation, with what functionality, etc.)

Creating Annotations

The following facilities were available for creating annotations: In the Wiki: (1) creating a Skipforward annotation (manually) in the Wiki. In Skipforward: (2) Manually by entering the name of a feature type on an item's page, (3) using the "Missing Features" recommender on an item's page, (4) using the item annotation recommender ("Recommended Features") on an item's page, (5) using the expert annotation recommender on user pages, (6) "replying" to an existing

feature by another user. Provenance information was crucial for later processing of the data collected and allowed creating statistics that were conceived only during running the experiments.

Provenance allows validating hypothesis H2.

System Changes During the Evaluation

While users were completing the evaluation, a number of bugs in Kaukolu and Skipforward were fixed, and many improvements were added:

- In Kaukolu, buttons to create annotations on touch screen devices were added, as the standard right click necessary to create annotations is not available on these devices.
- In Skipforward, the data visible to users was limited to the items that were part of the evaluation.
- In Skipforward, many UI changes were done, increasing visibility of the Skipforward trust mechanism, providing more navigation links, and improving handling of item reviews.
- In Skipforward, a view mode for the aggregated features was added that highlights cases of the current user's opinion being very different from the trust-weighted average of other users (see Figure 6.5; the normal view is shown in Figure 4.5). In this mode, in aggregated views, any statements by the current user are ignored; in the instances overlay, they are shown though, and if there is a strong divergence, an exclamation mark is added in the aggregated view.
- Additionally, a number of concurrency issues were fixed that lead to deadlocks.

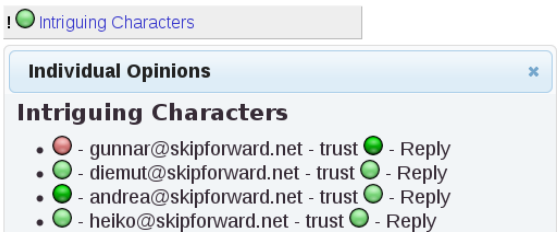


Figure 6.5.: Strong mismatch in weighted average and user statement.

Pre-Evaluation Questionnaire Results

State of the Art

This was an a priori questionnaire in which users gave their impression of the state of the art in existing services (e.g., Amazon) rather than the evaluated system. Topics were the user satisfaction and perceived importance of functionality such as item recommenders, search, and community- and trust-related features in systems that are in wide-spread use. Since the approaches evaluated in this chapter were born out of analyses of (shortcomings of) existing technology, application of collaborative approaches, and iterative improvements of existing technology, from a research perspective their relevance is easily perceivable;

Assessing Potential for Improvement

however, the notion of questionnaire is to assess the potential for improvement in the areas covered from an *end user perspective* instead of a research perspective. Note that this questionnaire can only give an abstract idea of the *potential* for improvement in the areas just mentioned; whether the approaches presented in this thesis result in an improvement cannot be answered by an a priori questionnaire. For this, a separate questionnaire has been designed, which evaluation participants were to fill out after the evaluation, having collected experience with the novel approaches. Fortunately, evaluating hypotheses H2 and H3 can be done based on data collected in the evaluation; the questionnaire results only serve as additional background.

Participants

The evaluation participants were asked to fill out the questionnaire shown in Appendix A. **15 users** participated in this questionnaire; seven of these had a knowledge worker background. In the answers, the people with the knowledge worker background did *not* form a cluster. Each question was optional, which explains differing numbers of answers in the following diagrams.

Recommender Performance

For presentation of the questionnaire results, diagrams that show importance and satisfaction of the feature in question have been chosen. Each of these diagrams covers one topic such as the user satisfaction concerning general recommender performance (see Figure 6.6). The exact statements that the users had to rate can be found in Appendix A; in this case, it was "The recommendations I get are personalized and useful for me". Users had to express their agreement with these statements on a discrete scale from 1 to 5 (1=strongly agree, 5=do not agree). Additionally, for each feature in question, they had to rate the importance they assigned to that feature. Importance was to be rated ignoring the state of the art. Consequently, it is possible that a user is strongly dissatisfied with the state of the art concerning a feature, but might assign that feature a very high importance, which would indicate a high potential for improvements in that area. In the table diagram, the cell color and number (size) signify the number of users giving the corresonding answer; for example, the cell with the dark background that is labeled with the number four represents the four

equal statements of four users, rating general recommender performance as in the middle ground, with an importance of recommenders slightly higher than that. It can be seen that importance is perceived as high in general, with good satisfaction, but room for improvement; no participant gave state of the art recommenders a very high rating concerning their satisfaction with it.

Figure 6.6.: Satisfaction with general recommender performance.

Tuning Recommenders

In Figure 6.7, the user perception concerning ways to tune recommendations, i.e., improving recommendation results by fine-tuning input criteria, is shown. Here, it is clear that there is very high potential of improving on the state of the art: Most users assign high importance to the feature but are rather unsatisfied with available implementations. This is expected since most available services have no or poor related features: for example, Amazon only allows removing recommended items and marking items that recommendations have been based on as unimportant. In purely Collaborative Filtering-based recommendations, this is about all one can do to support tuning recommendations.

Some participant comments concerning this question:

- *"The idea of being able to tune recommendations really appeals to me. At the same time, I'm a bit hesitant because a) I think I can't always say* exactly *why I like a story, b) I like getting surprised and would be worried that I'd narrow down my choice too much."*
- *"Giving feedback must not only be easy but the feedback* effectively *tune the recommendations (clicking feedback button should be "worth" it)."*

The second comment highlights the importance of *instant gratification*.

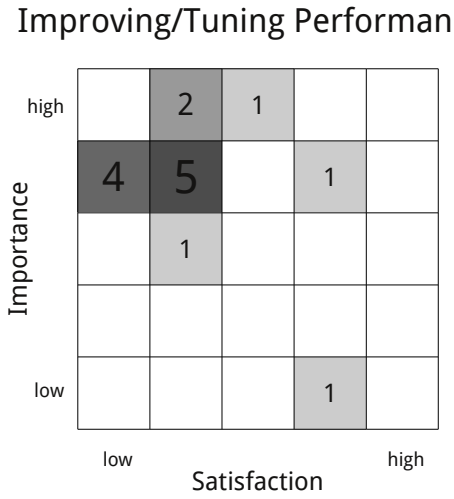


Figure 6.7.: Satisfaction with possibilities of tuning recommendations.

Trusted Reviews

In Figure 6.8, it is shown how easy users are able to find trusted reviews, and how important this is to them. It can be seen that most users find it important to be able to trust user reviews. Also, many users are not satisfied with the state of the art in this regard; finding trusted reviews should be made easier.

A participant commented concerning this question that deciding whether a review can be trusted can only be decided after reaching an own conclusion concerning the item in question. This highlights the lack of information concerning users in existing systems; often, it is even impossible to look up related information manually, e.g., looking up other reviews by the same user.

Similar Opinions

Related to the question of trust in reviews is how easy it is for users to find other users who are of a similar opinion as they are. First and foremost, this is important in areas with high subjectivity, e.g., concerning reviews or assessments concerning humor. Again, survey participants assigned this medium to high importance (Figure 6.9), but expressed their low satisfaction concerning this functionality in present systems.

Explanations

Most recommender systems give some kind of explanation along with the items they recommend. The explanation typically is a human-readable condensed version of the "data path" of the recommender process. In collaborative recommendations, a typical explanation is this:

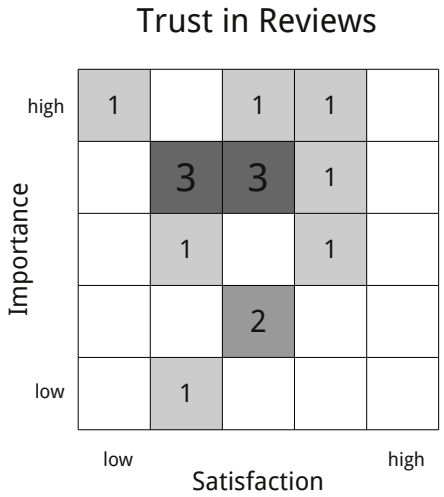


Figure 6.8.: Satisfaction concerning trust in user reviews.

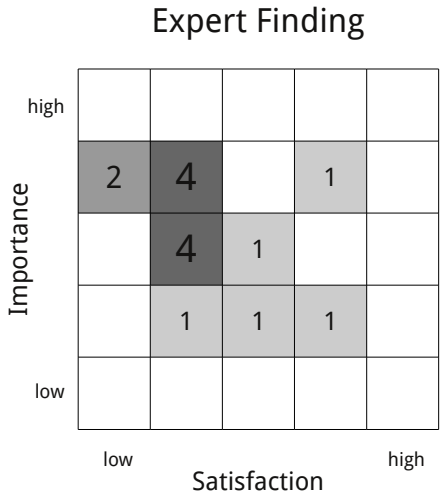


Figure 6.9.: Satisfaction concerning finding similar users.

> *"This item was recommended to you because it was often purchased by people who purchased items X, Y, and Z (that you purchased, too)."*

As seen in Figure 6.10, user satisfaction with the state of the art concerning explanations is mid to low. There is no clear consensus on the importance of explanations, possibly due to the wide variety of characteristics possible with regard to explanation implementations. However, while nobody assigned explanations low importance, several users did assign that feature high importance.

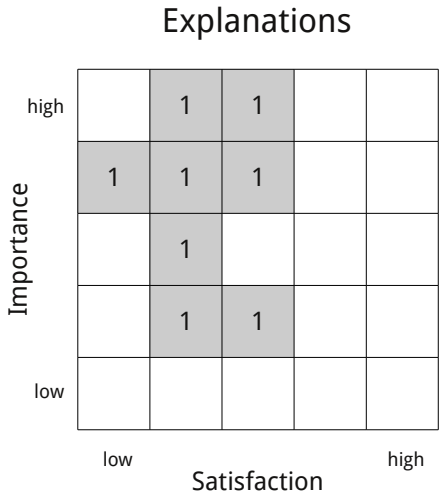


Figure 6.10.: Satisfaction concerning explanations in recommendations.

Goal-Directed Search

Figure 6.11 shows user satisfaction in goal-directed search, i.e., finding items by specifying a distinct profile of content-based features the items of interest is supposed to exhibit. General satisfaction is low; average perceived importance is middle to high, indicating a high potential for improvement of the state of the art. Since most platforms allow searching in item descriptions and technical metadata (author, release date, etc.) only, this is understandable.

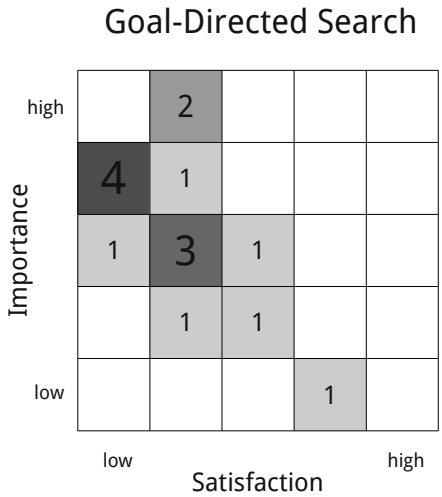


Figure 6.11.: Satisfaction concerning goal-directed search.

Drive-By Learning

When looking for relevant items, users often gain knowledge during the search process: for example, this can be because going through the options that are available highlights important aspects of the desirable item. Another possibility is that while browsing, the user learned something genuinely new about the domain. Figure 6.12 shows how well existing platforms support this form of "drive-by learning". Users indicate that they are moderately satisfied in this regard, but also mostly assign a moderate importance (three users assign a very high importance though). One hypothesis for explaining this is that users assume that when looking for an item of interest, a solid base knowledge about the domain is already present, and improving knowledge of the domain is of medium importance at best. Also, the original task specified was to find a relevant item; getting caught up in browsing additional information represents a distraction.

Some interesting user comments with regard to this question:

- *"Serendipity is nice but can get too much – see TV Tropes."*
- *"Actually not on the platforms such as Amazon. Such information is found on Wikipedia, Sci-Fi Websites or dedicated review sites."*

The first comment supports the hypothesis stated above saying that additional information could be seen as a distracting element. The second comment indicates poor integration of existing (sales) platforms with background knowledge repositories.

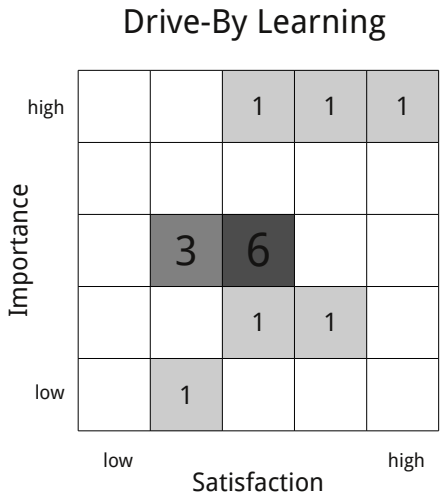


Figure 6.12.: Satisfaction concerning drive-by learning.

Item Summaries

Finally, users were asked how well typical item summaries work for them, i.e., how quickly they can find out what a story is about (Figure 6.13). Two groups are visible: One is satisfied with the state of the art, the other one sees room for improvement. Both groups assign this capability high importance on average. A hypothesis is that the group that is satisfied with the state of the art accepts that reading rather long descriptive texts is a necessity.

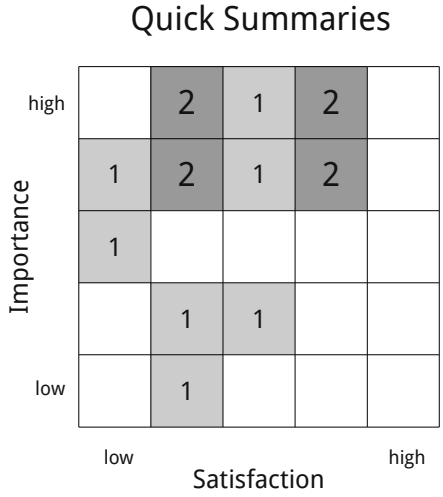


Figure 6.13.: Satisfaction concerning item summaries.

An interesting user comment was this:

> *"I think a look at the writing style of the author is most important for judging whether a book is suitable for me or not."*

This indicates that users are interested in specific facets of the item at hand.

Additionally, questions about annotations on paragraph level were asked. The participants assigned medium to low importance to this feature, presumably because this potentially distracts when reading and has little benefit in leisure reading. However, most people did indicate an interest in being able to globally search through any annotations created in such a way. This hints at synergy effects possible in multi-user scenarios.

General Statistics

The evaluation lasted about **ten weeks**, with **15 users** participating at the start. **Ten users completed the main part** of the evaluation which consisted of read-

ing the **25 core stories** and annotating them with the **31 required feature types**, writing a one-sentence review for each story, and assigning a general rating to each story. If the ten users, five had a knowledge worker background. Analysis of user correlation matrices showed that the people with a knowledge worker background did *not* form a cluster: There were both strong correlations between individual people from both groups as well as very weakly correlated people in the knowledge worker group. This was expected since correlations were built using the subjective story feature types which depend on each user's *taste* concerning short stories rather than skills in the domain of knowledge-based systems.

Analysis of Skipforward Annotations

Volume of the Data

Statistics

All in all, **9800 feature instances** were created, and **181 feature types** used. Only taking into consideration the data of the users who completed the evaluation, **9421 feature instance** were created and **177 feature types** used. Of these, **132 feature types were imported from DBTropes** using the DBTropes bot (see Section 5.7.3). About **300 comments** were present in the feature instances, plus about **250 short item reviews** (one review per user and item).

Annotation Sources

Annotations were created by different means. The main venue, due to the evaluation instructions, was creating annotations for the 31 required feature types using the "Missing Features" recommender that had been modified for this purpose, displaying required feature types first. Using other recommenders and feature types was left to user discretion.

- About **90 Skipforward wiki annotations** had been created.
- About **100 feature instances** were created **manually** by specifying a feature type on an item page in Skipforward.
- About **450 feature instances** were created using the **"Missing Features"** annotation recommender on item pages in Skipforward (ignoring instances of the 31 required feature types).
- About **650 feature instances** were created using the item annotation recommender on item pages in Skipforward.
- About **100 feature instances** were created using the annotation recommender that lists own items on feature type pages in Skipforward.

- About **10 feature instances** were created using the expert annotation recommender on user pages in Skipforward.

As can be seen, by far the most annotation instances were created using recommender functionalities. This is an indication that **hypothesis H2 holds**.

The Skipinions Ontology: Applicability and Confidence

One part of the claim of hypothesis H1 is that the Skipinions ontology is an intuitive way of annotating resources. Most of this claim has to be verified by explicit user feedback. However, one question that can be answered through analysis of the collected annotation data is whether the Skipinions approach of supporting truth values *and* user confidence has actually been used by the evaluation participants. If this was not the case, then the separation of truth values from user confidence is probably just complicating things, even if it is semantically sound.

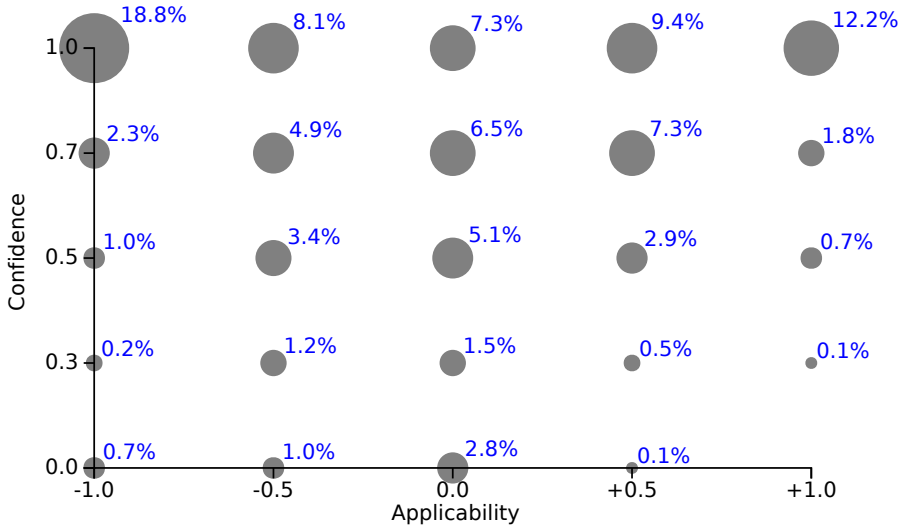


Figure 6.14.: Applicability/confidence value pairs used by User A.

Visualization

For judging this, the data collected in the evaluation was visualized using scatter plots. One such plot for one user can be seen in Figure 6.14. In these plots, the X axis denotes applicability, and the Y axis denotes confidence. While in theory the values are continuous, in practice they are discrete due to the design of the Skipforward user interface[5]. Multiple occurrences of one pair of applicabil-

[5]This is true for raw annotations entered by users directly. For aggregated values, this is not true anymore, but these are of no concern here.

ity/confidence values get aggregated. In the plots, the radius of each circle is proportional to the square root of the number of occurrences of the respective applicability/confidence value pair. The square root transformation was done since some values such as applicability $+1.0$ and confidence 1.0 as well as applicability -1.0 and confidence 1.0 occur far more often than other values naturally; without taking this into account, most of the plotted circle sizes would not be helpful (i.e., far too small). The relative number of occurrences is shown next to each circle to help interpreting the diagram. App/Conf

Figure 6.14 shows a quite extensive use of the decision space of the applicability/confidence approach by that user. As expected, the extreme values mentioned before show up in the upper corners. There seems to be a certain tendency to move to applicability 0.0 with decreasing confidence; this is intuitive, as, for example, the statement "The feature type occurs strongly, but I am not sure about this" rarely occurs in practice[6].

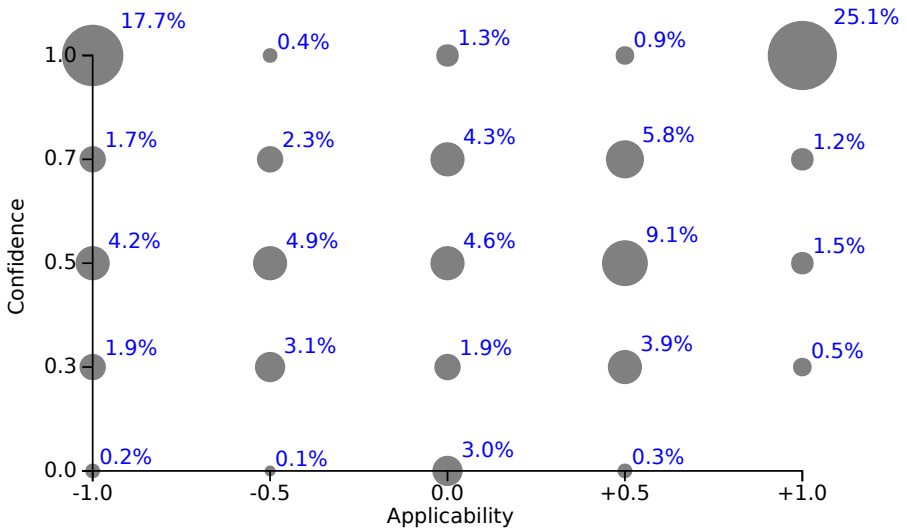


Figure 6.15.: Applicability/confidence value pairs used by User B.

Figure 6.15 depicts the usage of applicability/confidence by another user. Note that this user tends to use lower confidence in the less extreme applicability ranges, leading to a "V"-shaped plot rather than the "T"-shaped plot visible for User A.

Other users exhibit quite different behavior concerning their use of the applicability/confidence approach. Consider Figure 6.16; the user there almost ex-

[6]It *does* occur though; consider remembering a key scene of a story but not being sure if this was actually happening in *another* story, and memory just has failed.

clusively uses higher confidence values. It is interesting to see that in this case, the user seems more extreme in the decisions in general; not only are lower confidence values not used at all, but also the occurrences are even more strongly clustered in the upper extreme corners.

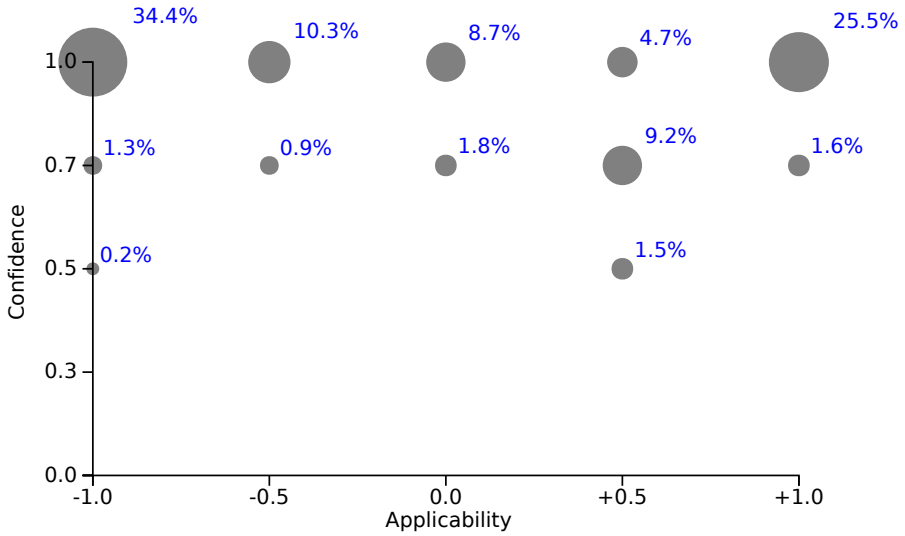


Figure 6.16.: Applicability/confidence value pairs used by User C.

All in all, this leads to the conclusion that while not all users make use of the complete decision space the applicability/confidence approach offers, some users do quite extensively.

Consequently, the applicability/confidence approach is valid (H1).

Additionally, in user interviews, the following user comments concerning applicability and confidence have been observed:

- *The confidence feature sped up creating annotations since uncertainness, this way, did not lead to pondering and losing time, but to just quickly creating an annotation with a low confidence value.*
- *In the applicability/confidence choice matrix, there were more options for confidence values than actually necessary.*

The Power of Subjectivity

Hypothesis H3 claims that differences in user opinions and building personalized views can improve recommendations. As outlined in the introduction of

this chapter, directly showing improvements in recommendations is difficult, as judgment of recommendation quality is very difficult. However, Skipforward's item recommendation approach is mostly based on content-based annotations. This, in turn, is based on the features items have been annotated with. Consequently, if it can be shown that the Skipforward approach leads to improved predictions concerning feature types, it follows that item recommendations based on these predictions will be better as well. This boils down to the following question:

> *"Are the aggregated user similarity-weighted views of Skipforward more accurate than non-weighted aggregated views?"*

Or, to phrase it differently: As outlined in Section 4.3.3, Skipforward can build aggregated views of other people's annotations by weighting these annotations with the user similarity for the feature type in question. In cases the current user did not annotate the item with the feature type in question, is the aggregated view (i.e., Skipforward's prediction of the user's opinion) more accurate than the simple mean of all other user's annotations? For cases such as generic item rating (the *Review* feature type in Skipforward), this is proven knowledge: collaborative filtering is based on this assumption. However, is this true for any feature type?

Special Case: Item Liking

One way to shed light on this question is running a cross-validation. In this case, this means that for each user and each item, an aggregated view of the features present in the system is built, ignoring the features of that user. Then, the error of the aggregated features (the prediction) to the actual features the user entered is calculated. This is done once for the aggregated view as described in Section 4.3.3 and once for an aggregated view with equal weighting for each user, i.e., *without* personalized views.

Cross-Validation

> Hypothesis H3 holds if the error for the weighted aggregated view is lower than the error for the non-weighted aggregated view.

In the case of the data collected in the 2013 evaluation, it has to be noted that a very difficult setting had been chosen:

- The required feature types were intentionally subjective.
- Some feature types had complex semantics.[7]

[7] For example *Would Make A Good Movie*: As indicated in the feature comments, some users typically assigned this a high applicability if they generally liked the story; others thought about whether the story would translate well into a movie.

- Some feature types had similar semantics.[8]
- The stories covered different genres and different moods.

Since for general item liking, the approach Skipforward takes is accepted in the form of *collaborative recommendations*, the results of the cross-validation for the *Review* feature type (representing basic item rating) can be taken as a baseline for error ranges to expect, as an indicator for anomalies, and as an indicator of the general difficulty of the task.

The complete statistical evaluation process was comprised of the following steps:

1. Run the cross-validation for the test item set and subjective features for different parameters of the Pearson correlation.
   As explained in Section 4.3.3, there are different parameters and aggregation functions involved in the personalization approach of Skipforward.
2. Investigate the behavior of the algorithm for different users and different feature types.

Evaluation Algorithm

The cross-validation was carried out as following. The evaluation software was run on the testing data which consisted of the annotations of the 10 users who completed the evaluation. Any redundant annotations (more than one feature instance for one item and feature type) were removed, and only the most recent ones kept[9]. The algorithm was a standard "leave one out" cross-validation.

Algorithm 6.4.1: CROSSVALIDATION()

for each $user \in evaluationParticipants$
  for each $item \in coreItems$
    $calculateCrossvalUserCorrelations(user, item)$
    for each $featureType \in subjectiveFeatureTypes$
      if($meanCorrelation(user, featureType)) >=$0.2
        and $\pi_2(f_b(user, item, featureType)) >=$0.5
        $weightedError+ = getWeightedError(user, item, featureType)$
        $nonweightedError+ = getNonweightedError(user, item, featureType)$

Some notes concerning this algorithm:

[8] Consider *Intriguing Plot* and *Convincing Plot*. While the difference between the two is obvious when thinking about it, when annotating in bulk the difference can be easily overlooked.

[9] The main algorithms in Skipforward already do this, but adding this functionality for the separate evaluation algorithms would have been a hassle and might have introduced bugs.

- *calculateCrossvalUserCorrelations(user, item)* calculates correlation values but *ignores* the annotations of *user* for *item* while doing so. This implements the "leave one out" approach.
- *meanCorrelation(user, featureType)* calculates the arithmetic mean of all user correlations to *user* with regard to *featureType*. Here, feature types with low overall correlation values are skipped.
- $\pi_2(f_b(user, item, featureType))$ is the confidence *user* assigned to the feature instance for *featureType* and *item*. This makes sure that no error is calculated for statements that have low confidence assigned.
- *getWeightedError()* calculates (or *predicts*, in this case) the weighted aggregated applicability for *user*, *item*, and *featureType*, then returns the difference of that to the applicability of the feature instance excluded in the cross-validation, multiplied with the confidence of that feature instance. *getNonweightedError()* uses the arithmetic mean when calculating the aggregated applicability.

Aggregated Weighted Error

The end result is the aggregated weighted error (of the prediction to the actual user-assigned applicabilities), which can be compared to the baseline represented by the naive non-weighted aggregation approach.

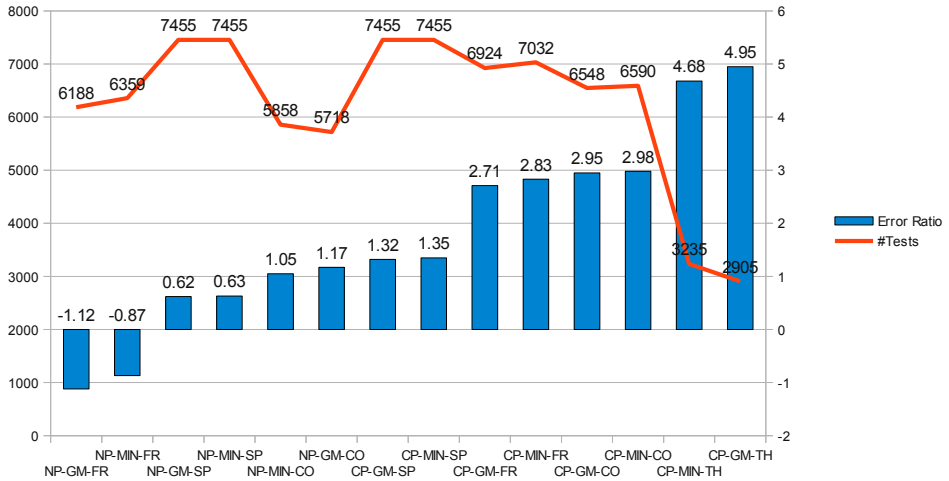


Figure 6.17.: Cross-validation error ratio and number of tests for different algorithm parameters.

Errors for Several Algorithm Configurations

In Figure 6.17, the results of the first step are shown.[10] On the X axis, the different algorithm configurations are listed[11]:

- NP: normal Pearson correlation. – CP: constrained Pearson correlation.
- MIN: minimum weight function. – GM: geometric mean weight function.

Similarity Projection Function

The last parameter is the similarity projection function $comp()$ (cf. Section 4.3.3) as shown in Figure 6.18. The FR function was included for testing whether including negative correlations in the weighted sum improves Precision. It has to be noted that the absolute value of FR is taken as weight in the weighted sum while the sign of FR is applied to the partial value in the weighted sum. The Y axis shows the error ratio, i.e., by how many percent the weighted approach outperforms the naive (non-weighted) prediction with regard to mean error.

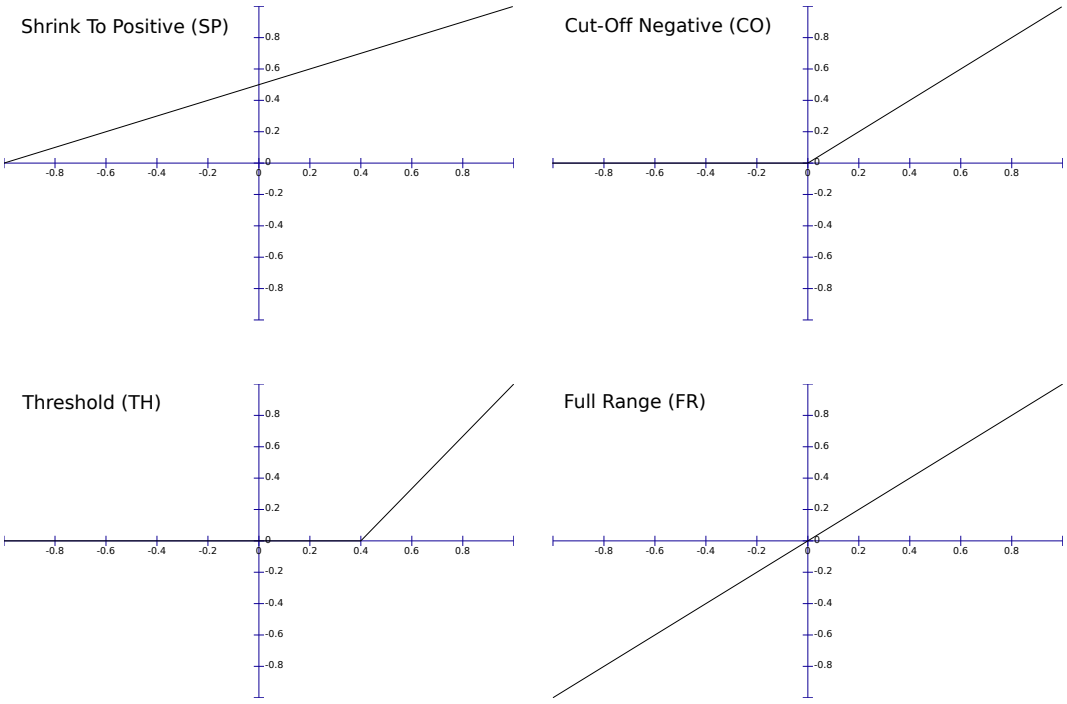


Figure 6.18.: Similarity projection functions.

[10] Computed using Java class `Eval2013` of Skipforward, SVN revision 1016 on `http://skipforward.opendfki.de/`.

[11] Two configurations are missing: The normal Pearson approach combined with the threshold similarity projection yielded correlation scores that were too low for the criteria in the testing algorithm.

Additionally, the number of individual comparison tests for each algorithm configuration is shown. A lower number means that for the configuration in question lower overall correlation scores resulted, which lead to more comparisons being excluded by the *meanCorrelation()* step in the algorithm, in the end leading to lower Recall.

The following conclusions can be drawn from the data in Figure 6.17.

- The constrained Pearson algorithm generally outperforms the standard Pearson algorithm, which is expected.[12]
- The min and geometric mean weight functions roughly behave similar.
- Making use of negative correlations (the FR similarity projection function) does *not* improve Precision.
- The threshold similarity projection function outperforms the other functions but at the cost of Recall (this is expected, as an increase in Precision typically results in worse Recall).

It has to be noted that in this scenario, "low Recall" means that the system detects that it has low confidence in the results but it still *could* calculate results if necessary, or fallback to the naive approach.

The roughly 5% improvement in mean error that the best configuration gives seem low at first. However, the noisy nature of the data has to be taken into account. Also, by heightening correlation thresholds it is possible to increase Precision further.

Detailed Analysis

In Figure 6.19, error ratios and the number of validation tests are shown per user[13] and feature type.[14] These have been computed with the threshold 0.4 similarity projection function, and ignoring all feature types with average correlation below 0.3: for example, for user A and feature type *Convincing Setting*, 25 comparisons of the weighted approach against the naive approach were done (which represents the full set of the 25 stories), and the mean error of the weighted approach was 29% better than the mean error of the naive approach.

[12] When looking at the raw data that consists of 160 test runs (one for each algorithm configuration and user), only three of the 80 test runs for the constrained Pearson configuration gave a negative error ratio, i.e., constrained Pearson giving worse results than the non-weighted approach. The normal Pearson algorithm performed worse than the naive approach in 30 of 80 tests.

[13] Note that user C has been omitted since that user had correlations above the threshold for two feature types only.

[14] The code for this can be found in Java class `Eval2013`, SVN revision 1018.

User A	#Tests	Ratio
BoilerplateStory	1	5%
ClearlyWritten	25	12%
ConvincingCharacters	25	14%
ConvincingPlot	8	9%
ConvincingSetting	25	29%
EasyToRead	24	0%
EntertainingHumor	22	-6%
IntriguingCharacters	25	8%
IntriguingSetting	5	-16%
MadeMeAngry	24	4%
MyNewFavoriteStory	19	6%
Review	25	-9%
TooManyInventedWords	23	32%
TooManyNames	23	5%
WorthReading	23	-13%
Total	**297**	**7%**

User B	#Tests	Ratio
ClearlyWritten	25	-3%
EasyToRead	24	6%
EntertainingHumor	25	14%
IntriguingCharacters	3	-16%
MyNewFavoriteAuthor	1	-20%
MyNewFavoriteStory	1	-32%
TooManyInventedWords	25	-3%
TooManyNames	24	8%
Total	**128**	**2%**

User D	#Tests	Ratio
ClearlyWritten	7	14%
ConvincingCharacters	23	14%
ConvincingSetting	23	28%
EasyToRead	17	4%
IntriguingCharacters	2	-15%
MadeMeAngry	22	-5%
TooManyInventedWords	24	7%
TooManyNames	2	-18%
Total	**120**	**9%**

User E	#Tests	Ratio
BoilerplateStory	2	10%
ClearlyWritten	25	-9%
ConvincingCharacters	25	7%
ConvincingSetting	2	-10%
EasyToRead	23	4%
EntertainingHumor	19	-9%
MadeMeAngry	25	17%
TooManyInventedWords	24	8%
Total	**145**	**4%**

User H	#Tests	Ratio
BoilerplateStory	15	7%
ClearlyWritten	20	5%
ConvincingCharacters	24	-2%
ConvincingPlot	23	16%
ConvincingSetting	19	15%
EasyToRead	23	2%
EntertainingHumor	19	7%
MadeMeAngry	24	5%
MyNewFavoriteAuthor	20	21%
MyNewFavoriteStory	20	17%
Review	17	-6%
SomethingNewUnderTheSun	16	-6%
TooManyInventedWords	22	36%
TooManyNames	22	1%
WorthReading	11	20%
Total	**295**	**10%**

User I	#Tests	Ratio
ClearlyWritten	2	9%
ConvincingCharacters	1	15%
EasyToRead	23	7%
EntertainingHumor	16	0%
MadeMeAngry	22	17%
MadeMeHappy	12	22%
MyNewFavoriteAuthor	21	10%
MyNewFavoriteStory	1	-1%
TooManyInventedWords	25	32%
TooManyNames	10	-5%
WorthReading	5	-5%
Total	**138**	**11%**

User F	#Tests	Ratio
ClearlyWritten	2	-4%
ConvincingSetting	2	-8%
EntertainingHumor	24	-1%
MadeMeAngry	24	4%
MyNewFavoriteAuthor	9	2%
MyNewFavoriteStory	11	3%
Review	3	15%
TooManyInventedWords	24	18%
TooManyNames	25	10%
WorthReading	3	-14%
Total	**127**	**5%**

User J	#Tests	Ratio
BoilerplateStory	14	10%
ClearlyWritten	1	-5%
ConvincingCharacters	14	5%
ConvincingSetting	6	-7%
EasyToRead	2	-2%
EntertainingHumor	19	4%
IntriguingCharacters	5	-22%
MyNewFavoriteAuthor	15	8%
TooManyInventedWords	20	12%
TooManyNames	22	34%
WorthReading	4	-1%
Total	**122**	**6%**

User G	#Tests	Ratio
BoilerplateStory	1	9%
ClearlyWritten	21	-2%
ConvincingCharacters	25	2%
ConvincingPlot	22	1%
ConvincingSetting	14	-11%
EasyToRead	12	8%
EntertainingHumor	25	1%
IntriguingSetting	25	15%
MadeMeAngry	25	18%
MyNewFavoriteAuthor	1	15%
MyNewFavoriteStory	1	5%
Review	25	16%
TooManyInventedWords	25	21%
TooManyNames	25	26%
WorthReading	14	4%
Total	**261**	**10%**

Figure 6.19.: Cross-validation results per user and feature type.

Feature types completely missing for individual users are due to low correlations or low confidence (see the cross-validation algorithm).

Several important observations can be made here.

- On average, the weighted approach outperformed the naive approach.
- For many feature types for which the weighted approach performs worse than the naive approach, actually very few test cases ran; this indicates noisy data and low confidence.
- For some feature types, the weighted approach performed very different for different users. Consider the feature type *Convincing Setting* for user G and user H.

- Many feature types did not meet the inclusion criteria for any user: for example, the feature type *Made Me Happy* is missing completely.
- In general, the approach worked better for users that were well correlated with other users: for example, error ratios are better for users that also have a high number of tests done in the cross-validation.
- The *Review* feature type only occurs for four users. For the other users, correlations and confidence was too low: for example, the feature type *Convincing Characters* met the inclusion criteria more often. Since the weighted aggregation approach is widely accepted for general liking (that *Review* represents), this is a strong indication that the approach works for other feature types as well.

To conclude the analysis, Figure 6.20 shows statistics per feature type.[15]

Statistics per Feature Type

	#Tests	Error Ratio	Avg. Corr	SumCT
BetterEnjoyedWithAlcohol	172	4.53%	0.27	340.66
BoilerplateStory	203	0.84%	0.45	637.23
ClearlyWritten	215	1.34%	0.60	982.43
CleverPlot	211	-0.26%	0.42	628.19
ConvincingCharacters	220	4.90%	0.53	893.86
ConvincingPlot	219	3.61%	0.44	745.52
ConvincingSetting	209	3.61%	0.54	807.55
ConvolutedPlot	217	5.64%	0.26	417.54
EasyToRead	217	0.78%	0.61	1050.92
EntertainingHumor	209	1.94%	0.60	971.95
GoodQuotes	199	-1.52%	0.34	441.92
GoodStoryTwists	206	3.87%	0.41	613.48
IntriguingCharacters	217	0.54%	0.49	836.97
IntriguingMood	216	7.81%	0.33	540.56
IntriguingPlot	216	2.70%	0.39	633.45
IntriguingSetting	217	2.27%	0.41	684.01
IntriguingWritingStyle	208	12.05%	0.18	278.48
MadeMeAngry	215	5.69%	0.53	920.38
MadeMeHappy	219	5.74%	0.42	725.35
ManyNewIdeas	202	-2.04%	0.31	417.96
MyNewFavoriteAuthor	203	2.56%	0.50	704.63
MyNewFavoriteStory	211	2.83%	0.48	771.50
PageTurner	212	-2.70%	0.33	519.06
Review	218	1.41%	0.50	895.50
SatisfyingEnding	212	1.29%	0.34	551.88
SomethingNewUnderTheSun	199	2.23%	0.41	513.12
StrangeMood	215	-0.13%	0.21	305.39
TaughtMeSomething	204	13.05%	0.24	360.92
TooManyInventedWords	212	8.49%	0.71	1236.76
TooManyNames	211	4.44%	0.61	1000.36
WorthReading	210	2.63%	0.53	816.98
WouldMakeAGoodMovie	207	5.45%	0.16	233.18

Figure 6.20.: Cross-validation results per feature type.

This cross-validation run was done using the cut off similarity projection function and without any correlation threshold in order to see detailed (but possi-

[15]The code for this can be found in Java class `Eval2013`, SVN revision 1019.

bly noisy) results for each feature type.[16] The number of tests column shows the number of cross-validations done. Maximum would be 225 (25 items times 9 users); the missing cases are due to users having assigned low confidence to their feature instances (see the cross-validation algorithm). The error ratio is the error improvement of the weighted approach compared with the non-weighted approach. The next column lists the average correlation of the feature type between users. The last column (SumCT) is the sum (over all tests) of the product of feature confidence and user similarity. This might give an indication of how trusted the results could be for the feature type in question.

The following observations can be made here.

- For almost all feature types, the weighted approach outperforms the non-weighted approach.
- In cases the non-weighted approach yields the better error, the difference to the weighted approach is small (maximum 2.7%).
- In all cases of the non-weighted approach providing a smaller error than the weighted approach, the average correlation is low (0.42 and lower).
- The *Review* feature type has a low error ratio of 1.41%. The applicability of this feature type is equivalent to general item liking and, thus, gives an indication of how well collaborative filtering would work in this scenario. Consequently, the low error ratio indicates a noisy data set or an otherwise difficult scenario.
- Some feature types that seem straightforward yield low error ratios (e.g., *Easy To Read*). This might be not because the weighted approach performed bad but because participants just agreed on the applicability of these types, leaving the weighted approach little room for improvement.

Conclusions

The following final conclusions can be drawn from the analysis.

1. The weighted approach results in improved Precision compared with the non-weighted approach for almost all cases and for almost all feature types.
2. The Precision of the weighted approach can be improved by excluding data with low confidence or correlation values.

   Since improved Precision for predictions concerning item features equals better input for recommenders, this proves hypothesis H3.

[16] Note that in this summary, user C has been left out due to the low correlations.

Interviews

After users read and annotated the stories, user interviews were done. These had an open format and, if appropriate, included working with the live system in order to explore and demonstrate extra functionality. In these interviews, the ten people finishing the evaluation were included. Five of these people had a knowledge worker background. While the general gist of the answers of these participants was in line with the people not having a knowledge worker background, the knowledge workers had occasionally more concise answers, and more concise expectations.

The results of the interviews can roughly be assigned to six groups.

Skipinions

The first group of statements is concerned with the **Skipinions** annotation approach. How easy was it to use? Were the semantics of confidence and applicability clear and useful? Was the cognitive load too high or workable? In the following, selected user statements are listed.

- *[The Skipinions approach] is good, clearly structured, and explained well. Initially, there was some ambiguity concerning applicability zero though.*
- *The applicability/confidence approach is really good. It makes entering information easier since you don't have to be absolutely sure about it.*
- *During annotation, the cognitive load for looking at the different aspects of an item is high and needs getting used to; assigning general liking only is easier.*
- *Over time, I got used to the approach, which made things much easier.*
- *General liking did not influence my handling of other positive feature types.*
- *I am not very knowledgeable in the domain of stories in the evaluation; therefore I often used low confidence when annotating.*
- *Some feature types were ambiguous, so sometimes the semantics I used them with changed depending on context.*
- *In the "Missing Features" recommender, a* Skip *button or something similar would have been good.*
- *Annotating while reading can be definitely helpful for some feature types such as* Good Quotes.

As visible, the approach did need some getting used to, but in the end, nobody had severe problems with the data model. For some cases such as users being unsure of their opinion, its high expressivity actually had benefits. This allows to draw the conclusion that **hypothesis H1 holds**.

Aggregated Views

The second group of statements is about **aggregated views** and the **trust model** coming into play there.

- *The aggregated view is interesting for comparing the own view with others.*
- *Being able to see user similarity is a very interesting feature.*
- *Skipforward is ideal for identifying people with overlapping interests. I'd actually like to meet some of the other people that participated in the evaluation.*
- *Functionality that highlights cases of clashing opinions for people with otherwise high correlation would be interesting.*

The functionality proposed in the last statement has been implemented prototypically during the evaluation.

Teamwork

The third topic is how the Skipforward approach influenced **working as a group**.

- *Using the system gave interesting insights concerning the point of view of other people, and allowed to broaden one's views and make one's judgments more objective.*
- *Ordering the list of people annotating the current item by the number of their annotations motivated me to annotate more.*
- *Seeing existing annotations motivated me to annotate more and look for fitting tropes in TV Tropes.*
- *Seeing power users at work was very motivating.*
- *On my own I would probably not create fine-grained annotations, but participating in a community might provide motivation for that.*
- *Being able to send passages to other users would be interesting.*

There were some unforeseen dynamics between user actions, and clearly users get more motivated when working as a team.

Recommenders/ Explanations

The fourth group of statements is about **recommenders and explanations** in recommendations.

- *The item recommender gave results that seemed fitting.*
- *The explanations for item recommendations are helpful.*
- *For item recommendation, separating subjective from objective features/tropes would be beneficial.*
- *The "Missing Features" recommender performed well.*
- *The expert annotation recommender is useful.*

Since there was little focus on item recommender functionality in the evaluation, the statements were quite general. The limited set of items visible diminished the usefulness of item recommendations. However, most users expressed an interest in seeing the data set growing. Concerning annotation recommenders, users were pleased in general. In combination with the collected quantitative data that indicated that annotations were mostly created using recommenders, it becomes clear that **Hypothesis H2 holds**.

Search and RSS

The fifth group of statements is about the **advanced search and RSS** features Skipforward provides.

- *Creating custom searches based on item profiles and being able to tune the search is very interesting.*
- *Advanced search gave interesting results, especially compared with fulltext search. I like this a lot.*
- *Custom search and associated RSS feeds are very useful. The push approach allows me to keep in touch with the system with low effort.*
- *For finding new items, I would look for the same author, use the similar item recommender, and do advanced search for mood-related feature types.*
- *Default searches that show up on the landing page would be great.*
- *I'd definitely like to subscribe a Skipforward "relevant items" RSS feed.*
- *Tunable search is cool.*

Some users were primarily interested in the mostly automated custom search functionality that builds search profiles based upon item liking ("recommender channels"). Other users wanted full control over the search process and were less concerned with the complex process involved in this. Since Skipforward,

in principle, allows combinations of both approaches, there is a lot of potential in this direction. It also became clear that the user interface is key to the user experience here.

TV Tropes

The last group of statements is concerned with the **TV Tropes integration**.

- *I used the TV Tropes integration a lot and imported a lot of tropes. One problem is the large set of tropes, but recommenders help.*
- *Finding a specific trope for a facet I currently have in mind is difficult. The trope hierarchy might be interesting to exploit here to reduce granularity.*
- *The short mouseover description for tropes was good, as was being able to navigate to TVTropes.org if needed.*
- *I almost found it easier to look up tropes in Skipforward than on TV Tropes, which would distract you.*

Overall, the TV Tropes integration was received very well. Finding a specific trope is still a challenge. While annotation recommenders help with annotating items, they are of limited use for this use case. DBTropes provides special functionality for this (e.g., intersection search that lets users look up a trope by specifying two items that feature that item), but this was not included in the evaluation. However, less complex approaches such as quick full text search in trope descriptions might help here as well.

Especially Noteworthy Statements

Finally, there were some quite strong and interesting statements that warrant separate mentioning.

- *Finding out how personal taste works and what makes something interesting to people is a hot topic. Skipforward has the data to analyze this.*
- *The feature types I find important personally were covered by the required feature types. I did not miss tagging.* — it has to be noted that this was voiced by a tagging enthusiast.
- *Being able to search for interesting items using fine-grained annotations is far better than standard collaborative recommendations.*
- *The Skipforward approach works.*

Final Questionnaire

After doing the interviews, users were asked to fill out a final questionnaire (Appendix D). Ratings were 1 (best/agree) to 5 (worst/disagree).

Results

The unanimous answer concerning the **item recommendations** was that there is high potential in the Skipforward approach (average rating: 1.3). Some users liked being able to look for specific tropes; others would like the system to build item profiles to look for them. The "recommendation channel" approach can do this, but its user interface needs to be improved.

The potential of Skipforward's **trust handling** received similarly high ratings (average: 1.85).

Explanations were deemed a useful feature (average: 2.4), but their presentation seemed to be not optimal at times. Better formatting and less clutter in the text descriptions should improve this.

The **expressivity** of Skipinions got good ratings (average: 1.7). As mentioned in the interviews, there was some confusion about applicability 0.0; improved paraphrases and paraphrase visibility (which was available as mouseover only) should help here.

Participants were divided concerning implementing Skipforward annotations on **paragraph level** in an eBook reader (average rating: 2.3). Several participants simply were not using eBook readers; some saw the need for paragraph-level annotations for specific feature types only. One participant mentioned that for data mining purposes the availability of the text of annotated passages would be very interesting.

The **TV Tropes integration** got a very good rating (average: 1.2). One participant mentioned that domain-specific integration with other sites such as book review sites would be beneficial as well; another participant mentioned that handling of feature type "ranges" (what types of items a feature type can be assigned to) might become a problem if handling more item types. Most annotation recommenders already take care of this problem (recommending only feature types that have been assigned to the current item type in the past), but this can be improved.

Continuous Usage

Finally, participants were asked if they could imagine continuing to use **Skipforward in the future**. The average rating was 2.4, with people divided into two groups (70% gave a 1 or 2 rating, 30% a 4). One comment was that the effort of annotating still needs to be reduced; the same person mentioned the benefits of eBook reader integration. One participant mentioned that allowing music and movies in Skipforward would be interesting. This could be done as the feature types for both domains are already available; for movies, DBTropes even provides a lot of instance information. Another comment mentioned the lacking

number of items present in the evaluation. This should be less of a problem once all items get unlocked for viewing and get annotated more uniformly. This ties in with some problems mentioned in interviews: A major problem in the short story domain is that in non-evaluation settings the stories are not available easily. Buying individual stories is usually not possible; they are available as bundles only typically. In Germany, the "Recht auf Privatkopie" (right to make copies for private usage) would theoretically allow giving copies of stories to friends (perfect in the Skipforward setting), but in practice often DRM makes this impossible.

Critical Remarks

Broad Selection of Stories

One problem is the small set of participants in combination with the broad selection of stories. In this setting, it is quite possible that there are no two users that have a shared taste with regards to the feature types to be graded. Using stories that are more similar would not have helped though; instead, likely only more noise would have been introduced.

Participants

The number of participants in the evaluation was relatively low; about half of the participants had an IT background. In general, in evaluations a high number of participants with a diverse background is desirable. As noted in the beginning of this chapter, some compromises had to be made to keep the effort and time requirements for the evaluation manageable.

The critical questions are: (i) What problems can arise with a relatively low number of participants, and (ii) What problems can arise if participants share an IT background.

Participant Count

Concerning the low number of participants, qualitative and quantitative evaluation results must be treated differently. Concerning *qualitative results* (e.g., questionnaire results), more participants mean more feedback in general, but there is a quick decline in useful feedback with increasing numbers of participants. As the usability researcher Jakob Nielsen observes [Nie12], for usability evaluations five participants are already enough in general. This is due to the fact that with increasing numbers of participants, the likeliness of each individual additional participant giving feedback that has not already been given before gets lower and lower. This is consistent with the observations made during the evaluation and during interviews: During later phases of the evaluation and for the interviews done last, little additional information was gathered. Consequently, 10 people participating in the evaluation was more than enough for gathering qualitative information.

Concerning *quantitative results* (e.g., the statistics used for investigating Hy-

pothesis H3), again, more participants would have meant more data to base the findings on. However, in this specific case of investigation, the potential problem with low numbers of participants would have been additional noise and, consequently, low correlation values and low improvement of error ratios. As discussed in the H3 investigation, this was not the case—or, to be exact, even with only 10 participants the measurable error ratio improvement was significant. To put this into additional numbers, the Student's t-Test[17] for Figure 6.20 results in a p-value of <0.001. This means that the likeliness of the sample observed in that figure is from a population with a mean of zero (i.e., no error ratio improvement is actually present: the observed improvements are due to random fluctuations in sampling) is less than 0.1%. It is interesting to note that this is despite Nielsen's statement that for quantitative measurements, 20 users are needed [Nie12]. The reason for this is that in the evaluation, in effect many small independent experiments are done, one for each feature type and user, instead of one per user. This results in high significance even with smaller participant numbers.

Participant Background

Concerning the participant background, again, qualitative and quantitative evaluation results can be treated separately. Concerning *qualitative results*, one observation was that people with IT backgrounds were able to give more detailed feedback concerning the workings of the systems, and suggested more specific improvements, which was actually beneficial in evaluation. Concerning *quantitative data* such as the aggregation of annotations created by the participants, no influence of the technical background on the user statements was expected. The reasoning here is that a participant's opinion concerning feature types such as *Intriguing Writing Style* are independent of their technical background knowledge and professional IT skills. This intuitive assumption is backed by a look at overall correlation matrices (see Appendix E): There were participants for all combinations of "(no) IT background" and "(no) SciFi fan", and no obvious overall correlations exist only between people with(out) IT background.

[17] A statistical hypothesis test—see `http://en.wikipedia.org/wiki/Student's_t-test`

V

Conclusion

CHAPTER 7
Outlook and Conclusion

The previous chapters examined how to facilitate collaborative semantic annotations and what advanced services can be built on these annotations. Special consideration was given to (i) a document-driven scenario that focused on annotating document repositories, and (ii) a resource-driven scenario that was concerned with annotating resources of any type such as books, or movies. Additionally, a Linked Open Data wrapping approach was presented whose output was used in both scenarios. The research focused on finding lightweight versatile annotation models, building goal-directed annotation recommenders, and creating personalized views from subjective annotations in multi-user environments.

In the conclusion of this thesis, an outlook over further applications that become possible with the technologies presented will be given. A short critical discussion of the approaches is given. Then, the major contributions of the thesis will be presented. The thesis ends with final remarks.

7.1. Outlook

Skipforward Approach in Everyday Apps

As indicated in the user interviews concluding the 2013 evaluation, an important point for user acceptance is making ontology-based annotation technology available in the applications of daily use: for example, for the book reading scenario, integration of Skipforward/Kaukolu technology in an eBook reader or eReader software would enable the user to annotate while using the application he typically uses for reading. Integration of further functionality – first and foremost social functionality, i.e., seeing other people's annotations and annotation summaries – would lead to a rich reading experience also facilitating connections between users of the system. Naturally, there would be a need to communicate with the other users. This ties in with Skipforward's initial focus as a semantically enabled forum (seen in Figure 7.1 at the lower right). Here, users can discuss the resources they know of, refer to specific feature types of these resources, and, while discussing, provide the system with rich metadata concerning conflicts of opinions. At the same time, the system can make use of the annotations generated and referenced by enhancing its search services,

Semantic Forum

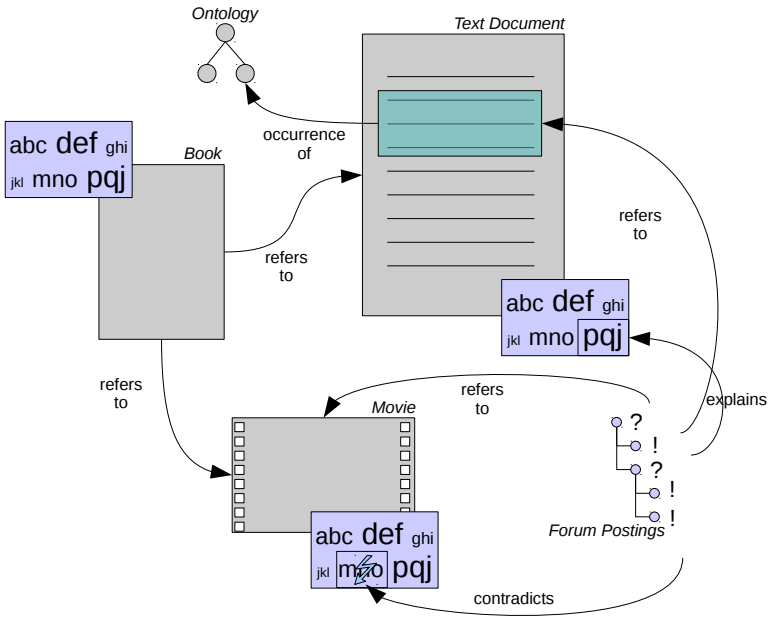


Figure 7.1.: A combined system using the Wiki, the resourced-based approach, and introducing forums.

for building aggregated views of the stored information, building concise summaries of relevant recent activity, recommending experts, and of course recommending items.

TV/Movie Applications

The possibilities of this technology are nearly endless: When DBTropes was introduced on the Linked Data mailinglists, people immediately thought about using its rich data in movie recommendations, or as a component in TV set top boxes that can recommend not only other similar movies based on the viewing habits of the current user, but also similar scenes. If the user created annotations while or after viewing movies, the system could facilitate discussion between users of the system (highlighting users with similar viewpoints, or alternatively highlighting opinion clashes). While creating annotations or complex interaction with the system while consuming resources is probably of limited interest to end users in leisure scenarios, even simple user interaction such as expressing "liking" while consuming a resource would be beneficial. In case indexed annotations for the resource are available (timestamped annotations for a movie or song; paragraph-referencing annotations for textual resources), the liking information can be used to derive more specific user profiles. Consider a user ex-

pressing liking in a movie scene that has been annotated with *Unflinching Walk*[1]; the system could infer then that the user likes (i) the *Unflinching Walk* trope, (ii) *Action Tropes*, (iii) *Action Movies*, as well as possibly (iv) *Badass Heros*[2].

Remix Culture

Other applications are possible. A limited kind of automatized remix culture implementation could be attempted: Imagine a software that carries out tasks such as "Create a remix of the best *Unflinching Walk* scenes in action films I and users with similar tastes like; start with *Animated Movies* first, then move to recent *Life-Action Movies*; use *Metal* music in the background, but *Instrumental* only please".

Understanding Trends

As indicated in the evaluation's user interviews, detailed information about entertainment media can lead to a more thorough understanding of (i) trends, (ii) preferences of consumers, (iii) the general workings of *liking*. Already the limited dataset collected in the evaluation gave interesting insights; not only did the prediction feature of Skipforward work reasonably well, but also inferences concerning the evaluation participants could be drawn: for example, native speakers of the English language had a quite different view on the *Good Writing Style* feature type than other participants. In the small dataset, this could be coincidence, but investigation with a larger more long-term dataset would give very interesting insights in this and other directions.

Data Model Improvements

Concerning the actual Skipforward system, several changes are worth pursuing. Additional functionality and most notably user interface changes could give quicker overviews of user clusters and hot topics. Concerning Skipforward's data model, a long-term change worth pursuing is inferring item types from the feature types associated with an item instead of explicitly assigning an item type. At the same time, more rigid handling of the domains of feature types needs to get implemented (i.e., what item types what feature types can get associated with). Concerning Skipforward's weighted views, alternatively to the user similarity per feature type approach using the Pearson correlation, predicting feature instances using other, more complex means is possible. Deriving feature instances by using combinations of the feature instances of other users *and* feature instances of other types should be possible in case the feature types are not completely independent. At the same time, rule-based inference should be possible, covering cases such as "The current user only shares opinions concerning liking with user B for items that feature the *Hard Rock* feature type.". This kind of domain-limited user similarity cannot be handled with the current Skipforward personalized views model.

[1] From TV Tropes: The hero walks away from the scene of previous action while an explosion takes place there; the hero does not even look back during this.

[2] These examples have all been taken from TV Tropes/DBTropes.

PIMO

The Skipforward/Skipinions approach can serve as a basis for other models such as the PIMO (see Section 2.3.1). The applicability/confidence notions of Skipinions would enable the PIMO to handle differing opinions of the participating users as well as allow users to assign individual confidence values to their statements. The Skipinion provenance approach allows simple handling of multi-user scenarios while, at the same time, keeping clear boundaries with regards to individual user's datasets.

7.2. Critical Remarks

While **Semantic Wikis** are a powerful tool, due to their high degree of formality and the large amount of background knowledge needed by the user, their full potential can only be used in circles of power users. However, due to their collaborative nature, this workload can be distributed relatively easily over the user base. Another aspect is that refactoring (changing a Wiki's structure; cleaning up texts that have evolved over time, and adjusting structure) gets more difficult the more additional functionality a Wiki provides: A user that is refactoring a text needs not only to keep track of the text flow but also semantics when using Semantic Wikis. With Kaukolu's approach of detached annotations, when moving around or deleting large amounts of texts, annotations can become orphans, losing their association to the text. This can get tackled using additional housekeeping functionality (e.g., a tool that informs the user of orphaned annotations), but again, this is additional complex workload.

For **Collaborative Annotation Systems**, the trust-/competence-based weighting approach gives great potential for improved personalized views. However, this comes at the cost of higher system complexity, both concerning user interfaces and computation load for the system in the background. At the same time, requirements and potential of the approach is different for different use cases: Consider (i) a setting with 10 users in a small workgroup, and (ii) a setting with thousands of users. In the first setting, the way Skipforward presents and handles trust is workable and gives users valuable information about their peers. In the second setting, this is not possible easily anymore: For example, presenting a list of all other users that have expressed an opinion regarding a feature type along with trust values is not feasible for thousands of users. Not only would the list be too long; the usernames presented will not mean anything to the user looking at the list. Additional facilities to limit the output of the system would need to be put into place; however, this is additional complexity, likely resulting in non-determined behaviour from the user's point of view. Consequently, computation of personalized views is likely more

important in use cases with many users, while the availability of trust values is already beneficial in scenarios with few users.

The **Online Wrapping Approach** for Linked Open Data provides a fast way of making data previously opaque to machines available for machine processing. In case the wrapped website changes its basic structure, though, the approach can fail, and major adjustments to the wrapping machinery might get necessary. Consequently, as with all complex toolchains that work continuously, checks and test cases need to be built that notify the people running the wrapper of changes that possibly broke the extraction process.

7.3. Conclusion

In the following, the contributions of the work are listed along with their relation to the research hypotheses presented in Section 1.2.

Semantic Wikis

For the **Semantic Wiki** domain, *Kaukolu Wiki* was developed. It represents a prototype that investigates novel ways of creating and making use of agile semantic annotations.

The **annotation ontology** used by Kaukolu allows annotating documents without changing them which is not possible with most other Semantic Wikis. It allows assigning arbitrary instances of concepts described by domain ontologies to parts of text documents. Furthermore, Kaukolu **combines several types of annotations** and annotation metadata to allow ways of searching and querying that previously, without these facilities of annotations, were not possible: for example, systems that store annotations directly within the Wiki markup cannot represent automatically generated attention annotations or reliably keep provenance information for the annotations.

Fine-grained document annotations that can overlap and reference each other can be complex to handle. Therefore, the system assists users with special **annotation recommender functionality** while reading and annotating documents. This functionality makes use of structural document information as well as restrictions given by the annotation ontologies used.

To demonstrate the enabling technologies introduced in the approach, several use cases have been presented that highlight the features of the system. These use cases cover several domains with different requirements that have been outlined in this thesis.

Collaborative Annotation Systems

In the **Collaborative Annotation Systems** domain, the *Skipforward* system was developed. It implements a collaborative ontology-based annotation system based on a **lightweight annotation ontology** (H1).

Skipforward focuses strongly on multi-user usage and provides **fully personalized views** of the data contained in the system by means of its **competence metric** and associated weighting of user opinions (H3). In contrast to existing systems, it implements a **unique hybrid recommender approach** that allows functionality not possible with other systems such as fine-grained explanations and fine-tunable parameters during the recommendation process. The annotation recommenders introduced with Skipforward are specifically tailored for its annotation model and target not only improving item annotations and recommendations but also **improvements in expert recommendation**.

A use case in the domain of book annotation has been presented. Both the user acceptance of the different annotation recommenders as well as the improvement of recommendations using the Skipforward annotation model has been **evaluated** (H1, H2, H3). The collected dataset is freely available using Linked Data APIs.

Linked Open Data

Several approaches for wrapping existing Websites and serving Linked Open Data were presented. As a proof of concept for the **online wrapping approach**, *DBTropes* was developed. It represents an online wrapper that transforms information stored in a traditional Web-based system into Linked Open Data. For representation of the original data, the Skipforward base ontology is used (H1).

The online wrapping approach of DBTropes keeps its available information up to date and in synchronization with the parent service. For the TV Tropes domain, it is shown that **high Precision** concerning data quality can be reached by verifying the data extracted against domain restrictions. Additionally, manual improvements in Precision can be achieved using its **online data housekeeping functionalities**. The approach has been shown to be able to handle **large volumes of information and updates**. The DBTropes data is freely available using Linked Data APIs and represents one of the bigger Linked Data sources available on the Web. Furthermore, the DBTropes data is used within the Skipforward system directly (as well as a number of other systems), effectively bringing together different communities.

An **evaluation** showed the high Precision the approach is able to achieve (H4).

7.4. Final Remarks

In this thesis, two models for representing annotations have been developed and demonstrated using prototype systems. One model focuses on a document-based scenario; the other focuses on a resource-based scenario. In the document-based scenario, being able to make use of user attention information and user context information enhances retrieval and personalization tasks; the resource-based scenario handles personalization tasks, too, but focuses on deriving user similarity values from overlapping annotations, which are much more numerous in this scenario than in the document-based scenario.

In the evaluation, both annotation models got combined, complementing each other. The evaluation showed that the proposed personalization approach based on weighting user statements by user similarity outperforms the non-weighted approach for almost any type of annotation facet. Therefore, the approach is well suited for handling and aggregating user-contributed annotations in multi-user scenarios. It will be interesting to see the approach applied to more use cases, some of which have been outlined in the previous sections.

APPENDIX A
Pre-Evaluation Questionnaire

This is the questionnaire the evaluation participants had to fill out online before doing the main evaluation. For every question, both the degree of of agreement and importance/weight of the question was requested.

Scenario

Imagine you are an avid scifi/fantasy reader. To find stories that are worth reading, you use services such as Amazon's "People who have bought this also bought" recommender service, its top seller lists, and user reviews.

Questions

Most questions come in tandem with a question that allows you to express how important the previously presented feature would be to you.

1. **The recommendations I get are personalized and useful for me.**
   "useful" as in "I find something new", "something that fits my taste", etc. - keep in mind to answer these questions as a scifi/fantasy fan (see the scenario description).
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Personalized useful recommendations would be very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

2. **It is easy for me to "tune" my recommendations.**
   For example, as in "That one recommendation isn't too bad, but I like books with more action in them, please consider this in future recommendations".
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Being able to tune recommendations would be very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

3. **I can trust the reviews I find.**
   I.e., for the reviews/reviewers I find I can easily see whether I will agree with its statements ("This review looks legit.").
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Trust in reviews is very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

4. **Finding out WHY something was recommended to me is easy, and the explanations given are helpful for me.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Explanations in recommendations are very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

5. **It is easy for me to find books playing with a specific idea.**
   I.e., "I want to read a scifi story mostly taking place off-earth"
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Being able to find these books easily is very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

6. **Comparing items/books is easy.**
   E.g., when deciding which of two books to buy, the seller's platform makes it easy for me to compare them.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Being able to compare items easily is very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

7. **Finding experts who write reviews that I agree with is easy.**
   This is about finding people, not about the general quality of all reviews/user opinions.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Finding experts easily is very important to me.

Select a value from a range of 1, strongly agree, to 5, strongly disagree.
Comment: _____

8. **I learn a lot while looking for interesting new books/items.**
   E.g., while browsing for interesting new stuff, I often stumble about interesting background knowledge of the domain.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Learning while browsing is very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

9. **I can easily re-find a specific story I read.**
   E.g., you read/bought a book years ago, don't remember its title anymore, but want to recommend it to a friend.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.

   Re-finding stories easily is very important to me.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   Comment: _____

10. **It is easy to quickly see what a book/story is about.**
    I.e., its setting, storytelling style, etc., is visible in the supplied information.
    Select a value from a range of 1, strongly agree, to 5, strongly disagree.

    Quick summaries are very important to me.
    Select a value from a range of 1, strongly agree, to 5, strongly disagree.
    Comment: _____

11. **I want to be able to mark passages while reading.**
    ...or create other types of annotations.
    Select a value from a range of 1, strongly agree, to 5, strongly disagree.
    Comment: _____

12. **I want to re-find marked passages in a global search interface for the whole of my library.**
    E.g., "Show me all passages that I marked with the word 'scary'."
    Select a value from a range of 1, strongly agree, to 5, strongly disagree.
    Comment: _____

APPENDIX B
Evaluation Instructions

These are the evaluation instructions made available to evaluation participants in Kaukolu.

- Fill out the accompanying questionnaire. [link]
- Go to [link] and log in using your Skipforward credentials. [...]
- Firefox is recommended for accessing this wiki. [...]
- Browse around a bit. Note there are tooltips for most things: move the mouse cursor over something and just wait a bit without clicking.
  - For example, look at this story and these features/tropes.
  - This is also a good point in time to call Malte who will give you a short walkthrough of the Skipforward system.
- Come back here.
- On this wiki's main page (see the top link in the left menu), there is a list of short stories. Please read at least 25 of them over the course of the next weeks, if possible online in the wiki.
- The following few points are optional. If you are in a hurry, skip to "switch to Skipforward".
  - Ignore other people's annotations. You can hide them by playing with the "author" drop-down box at the top.
  - If reading online is not an option, see the PDF with all the stories here [link]. But do read the stories with a marker in hand and mark any passages that (probably) exhibit some trope (or a subjective story feature, see below).
  - Please read the stories in the order indicated on the wiki main page.

- While reading online, you can use the mouse to select text, then... a) right-click and create an annotation or b) click one of the buttons on the top right corner.
- Create Skipforward annotations ("Create Skipforward Annotation") if possible; if you cannot (e.g., you do not know the appropriate Feature Type's label), just create freetext annotations ("Create Annotation", "Comment", [...]) for the time being, and give a comment of what you observed at that position of the story.
  - The circle encodes applicability/confidence. Red=This does not apply, green=this applies. Big=I'm sure, small=I'm not sure. Click and drag to modify. Mouse hover to get a paraphrase.
  - You need to be logged in in Skipforward [...] when creating Skipforward annotations in the wiki.
- For each story, after reading, please **switch to Skipforward**, and continue annotating there. Have a look at the annotation recommenders (Missing Features and Recommended feature types on Item pages).
  - Please annotate at least all the Subjective Story Features [link] for each story. These get listed first in the "Missing Features" section of an item's info page (linked to on the Main page of this wiki). Hovering the mouse cursor over "Why?" (no click!) gives a "This is a required feature" text box for these.
  - The circle encodes applicability/confidence. Red=This does not apply, green=this applies. Big=I'm sure, small=I'm not sure. Click, hold and drag or click and select to modify. Mouse hover to get a paraphrase.
  - "Meh, I do not have any opinion concerning this" $\Rightarrow$ select yellow circle with minimum size.
  - If you are not sure about something, just select low confidence/small circle, but please keep annotating.
- When you have enough of reading and annotating (and finished the first 25 stories), contact Malte.
- Malte will also explain recommendation channels and Advanced Search. Play with these functionalities for a bit.
- The evaluation will close with a last feedback questionnaire.

APPENDIX C
Stories and Feature Types

Short Stories

iRobot by Guy Haley (1200 words, Interzone 244)
The Angel at the Heart of the Rain by Aliette de Bodard (1705 words, Interzone 246)
Railroad Angel by Gareth L. Powell (1811 words, Interzone 241)
The Core by Lavie Tidhar (3605 words, Interzone 246)
A Flag Still Flies Over Sabor City by Tracie Welser (3704 words, Interzone 244)
Sentry Duty by Nigel Brown (4094 words, Interzone 246)
Paskutinis Iliuzija (The Last Illusion) by Damien Walters Grintalis (4100 words, Interzone 245)
Malak by Peter Watts (4382 words)
The Genoa Passage by George Zebrowski (4515 words, Interzone 244)
The Remembered by Karl Bunker (4619 words, Interzone 242)
The Philosophy of Ships by Caroline M. Yoachim (4812 words, Interzone 243)
Triolet by Jess Hyslop (4875 words, Interzone 246)
Thesea and Astaurius by Priya Sharma (4932 words, Interzone 246)
Needlepoint by Priya Sharma (5120 words, Interzone 242)
Cat World by Georgina Bruce (5201 words, Interzone 246)
Ship's Brother by Aliette de Bodard (5500 words, Interzone 241)
Wonder by Debbie Urbanski (5502 words, Interzone 242)
Beyond the Light Cone by C. W. Johnson (5605 words, Interzone 242)
Lady Dragon and the Netsuke Carver by Priya Sharma (5898 words, Interzone 243)
The Moral Virologist by Greg Egan (5918 words)
Invocation of the Lurker by C. J. Paget (5984 words, Interzone 241)
Bit Rot by Charles Stross (6459 words)
Extracts From the Club Diary by Charles Stross (6632 words)
A Career in Sexual Chemistry by Brian Stableford (7172 words)
Understand by Ted Chiang (13344 words)

Subjective Feature Types

Better Enjoyed With Alcohol – Being drunk when reading this would have improved things a lot.
Boilerplate Story – I think I have read similar stories many times.
Clearly Written – I think the story has been written in a clear style. It might still be difficult to understand or use complex sentences etc., but there's no ambiguity.
Clever Plot – The plot is quite well thought out.
Convincing Characters – I found the characters in the story believable; they do not feel overly constructed.
Convincing Plot – I found the story plot believable (given its setting and genre); it did not involve much handwaving.
Convincing Setting – I found the story setting believable (given its genre); it did not feel contrived.
Convoluted Plot – I found the plot quite convoluted. Perhaps less could have been more in this case.
Easy To Read – I think the story is easy to read: Straightforward language, simple story concepts, clear storyline.
Entertaining Humor – I found the humor in the story entertaining; it worked well for me.
Good Quotes – I think this story has a lot of good quote material.
Good Story Twists – I liked the story twists.
Intriguing Characters – I found the characters intriguing. They might not be very believable but who cares.
Intriguing Mood – I found the mood intriguing.
Intriguing Plot – I found the plot intriguing. It might not be very believable but who cares.
Intriguing Setting – I found the setting intriguing. It might not be very believable but who cares.
Intriguing Writing Style – I found the writing style intriguing.
Made Me Angry – Reading this story made me angry.
Made Me Happy – Reading this story made me happy.
Many New Ideas – The story presented a lot of new ideas to me.
My New Favorite Author – This is an author I have to keep an eye on.
My New Favorite Story – This story is one of my new favorites.
Page Turner – I could impossibly stop reading.
Satisfying Ending – I found the story ending satisfying. It might not have been happy or closed, but it certainly felt right.
Something New Under The Sun – Of at least one major thing in the story I'm quite

sure it's something genuinely new.
Strange Mood – The story is told in a strange mood.
Taught Me Something – This story taught me something new.
Too Many Invented Words – In my opinion, the story uses far too many made-up words.
Too Many Names – The story drowned me in names of people, places, etc.
Worth Reading – I recommend the story, even if it's possibly a bit of work to read it.
Would Make A Good Movie – This story would make a good movie.

Review – Put a plaintext review in the comment field. Applicability +1: Recommendation. Applicability -1: Avoid the associated Item.

APPENDIX D
Post-Evaluation Questionnaire

This is the final questionnaire evaluation participants were asked to fill out online after doing the main part of the evaluation, and after user interviews.

1. **Personal item recommendations using the Skipforward approach have a lot of potential.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   What would you hope for, what facet of recommendations do you find most interesting, etc.

2. **I see a lot of potential concerning trust handling in Skipforward.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   Again, what functionality could you imagine; what was missing.

3. **I liked the explanations ("WHY?" mouseovers) in Skipforward.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   What could be improved, what was missing.

4. **The range of statements I was able to enter in Skipforward was big enough.**
   "Strongly disagree" would mean there were a lot of statements you would have liked to enter in the system that it didn't allow you to enter.
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

Comment: _____
What could be improved, what was missing.

5. **I would like to have Skipforward-like annotation functionality on paragraph level in an eBook reader.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   ...what could you imagine happening? Think social media, recommendations.

6. **I liked the TV Tropes integration in Skipforward.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   What could be improved, what was missing. Any other source of feature types or integration with other websites that could be interesting?

7. **I can imagine using Skipforward in the future.**
   Select a value from a range of 1, strongly agree, to 5, strongly disagree.
   _____

   Comment: _____
   What keeps you from continuing using it? What feature is missing to convince you? What domain other than books would you like covered in Skipforward? Music? Films? Games? Politicians?

8. **Final Comments** _____

APPENDIX E
User Correlation Matrices

In the following, four example user correlation matrices are shown, one for each combination of "is (no) SciFi fan/has (no) IT background".

The "Total" column *printed in bold gives an indication of how well the user is correlated with all other users concerning the feature type in question; the "Total"* row *gives an indication of how well the user is correlated with others users all in all.*

To give an example, in Figure E.1 the user "blends in best" concerning the opinion with regard to "Too Many Invented Words"; and in general, the user agrees most with User 8 and least with User 3.

u01--	TOTAL	u02I-	u03--	u04--	u05--	u06IS	u07IS	u08IS	u09-S	u10IS
Better Enjoyed With Alcohol	**3.26**	0.48	0.21	0.14	0.29	0.52	0.47	0.35	0.5	0.3
Boilerplate Story	**4.84**	0.48	0.31	0.45	0.5	0.62	0.71	0.73	0.36	0.68
Clearly Written	**5.75**	0.74	0.59	0.86	0.82	0.46	0.72	0.68	0.39	0.48
Clever Plot	**4.05**	0.43	0.47	0.5	0.35	0.51	0.64	0.7	0.02	0.44
Convincing Characters	**6.02**	0.59	0.77	0.91	0.75	0.24	0.78	0.85	0.57	0.55
Convincing Plot	**4.66**	0.41	0.35	0.74	0.62	0.54	0.75	0.86	-0.02	0.42
Convincing Setting	**5.47**	0.35	0.28	0.94	0.63	0.79	0.6	0.85	0.39	0.65
Convoluted Plot	**3.5**	0.46	0.37	0.01	0.04	0.59	0.42	0.73	0.66	0.23
Easy To Read	**6.21**	0.83	0.66	0.86	0.76	0.57	0.59	0.73	0.6	0.61
Entertaining Humor	**6.37**	0.73	0.7	0.45	0.69	0.87	0.79	0.81	0.6	0.73
Good Quotes	**3.82**	0.37	0.22	0.23	0.36	0.56	0.55	0.52	0.35	0.66
Good Story Twists	**4.25**	0.33	0.15	0.72	0.41	0.34	0.7	0.49	0.43	0.68
Intriguing Characters	**5.52**	0.78	0.61	0.73	0.73	0.47	0.72	0.6	0.22	0.68
Intriguing Mood	**4.08**	0.64	0.28	0.6	0.6	0.01	0.58	0.74	0.14	0.49
Intriguing Plot	**3.95**	0.52	0.12	0.49	0.4	0.31	0.72	0.65	0.17	0.57
Intriguing Setting	**4.74**	0.49	0.06	0.58	0.58	0.68	0.83	0.61	0.33	0.58
Intriguing Writing Style	**0.48**	0.29	-0.12	0.43	-0.42	-0.46	0.22	0.52	-0.1	0.13
Made Me Angry	**5.19**	0.11	0.54	0.72	0.63	0.65	0.67	0.61	0.78	0.47
Made Me Happy	**3.65**	0.39	0.48	0.24	0.55	0.34	0.53	0.43	0.35	0.34
Many New Ideas	**2.91**	0.08	0.22	0.48	0.15	0.14	0.33	0.55	0.49	0.48
My New Favorite Author	**4.65**	0.57	0.32	0.22	0.42	0.59	0.61	0.6	0.59	0.72
My New Favorite Story	**5.03**	0.55	0.35	0.16	0.67	0.75	0.66	0.72	0.63	0.54
Page Turner	**4.26**	0.49	0.45	0.35	0.24	0.36	0.74	0.72	0.4	0.52
Review	**5.5**	0.65	0.35	0.57	0.65	0.64	0.78	0.67	0.65	0.54
Satisfying Ending	**4.13**	0.53	0.44	0.5	0.39	0.43	0.58	0.81	0.13	0.31
Something New Under The Sun	**4.64**	0.65	0.38	0.37	0.39	0.32	0.79	0.62	0.6	0.51
Strange Mood	**2.13**	0.06	0.48	-0.09	0.02	0.42	0.19	0.45	0.39	0.21
Taught Me Something	**2.28**	0.24	0.28	-0.01	0.07	0.44	0.36	0.28	0.44	0.17
Too Many Invented Words	**7.23**	0.7	0.31	0.88	0.9	0.84	0.93	0.95	0.95	0.78
Too Many Names	**6.37**	0.73	0.58	0.66	0.45	0.75	0.86	0.75	0.77	0.82
Worth Reading	**5.92**	0.65	0.47	0.63	0.55	0.78	0.76	0.72	0.68	0.68
Would Make A Good Movie	**0.57**	0.54	-0.26	-0.11	-0.05	-0.16	0.37	0.16	-0.22	0.31
TOTAL	**141.41**	**15.84**	**11.42**	**15.18**	**14.13**	**14.94**	**19.95**	**20.45**	**13.22**	**16.27**

Figure E.1.: Example user correlation matrix: No SciFi fan, without IT background.

u02I-	TOTAL	u01--	u03--	u04--	u05--	u06IS	u07IS	u08IS	u09-S	u10IS
Better Enjoyed With Alcohol	**3.25**	0.48	0.39	0	0.39	0.44	0.61	0.31	0.5	0.11
Boilerplate Story	**2.89**	0.48	0.01	0	0.67	0.24	0.3	0.29	0.51	0.39
Clearly Written	**5.95**	0.74	0.46	0.81	0.69	0.73	0.64	0.67	0.51	0.69
Clever Plot	**3.61**	0.43	0.36	0.14	0.16	0.29	0.43	0.65	0.51	0.64
Convincing Characters	**4.36**	0.59	0.59	0.71	0.51	0.07	0.49	0.53	0.42	0.45
Convincing Plot	**3.54**	0.41	0.12	0.35	0.17	0.34	0.57	0.62	0.39	0.56
Convincing Setting	**3.62**	0.35	0.36	0.3	0.3	0.45	0.48	0.38	0.47	0.53
Convoluted Plot	**1.22**	0.46	-0.05	-0.2	0.21	0.34	0.2	0.29	0.08	-0.1
Easy To Read	**5.8**	0.83	0.44	0.75	0.72	0.72	0.61	0.7	0.53	0.5
Entertaining Humor	**5.22**	0.73	0.44	0.18	0.58	0.74	0.61	0.59	0.55	0.8
Good Quotes	**2.6**	0.37	0.53	0.27	0.24	0.19	0.15	0.46	0.31	0.09
Good Story Twists	**3.56**	0.33	0.21	0.12	0.21	0.4	0.66	0.48	0.54	0.62
Intriguing Characters	**4.95**	0.78	0.33	0.51	0.56	0.31	0.63	0.6	0.59	0.62
Intriguing Mood	**3.4**	0.64	0.07	0.23	0.51	-0.06	0.56	0.38	0.52	0.56
Intriguing Plot	**3.7**	0.52	-0.07	0.18	0.27	0.52	0.64	0.62	0.43	0.58
Intriguing Setting	**3.24**	0.49	0.14	0.22	0.32	0.53	0.68	0.15	0.25	0.47
Intriguing Writing Style	**1.77**	0.29	-0.04	0.07	-0.09	-0.13	0.65	0.24	0.21	0.57
Made Me Angry	**2.02**	0.11	0.35	0.11	-0.03	0.38	0.37	0.57	0.02	0.14
Made Me Happy	**4.03**	0.39	0.48	0.35	0.31	0.54	0.51	0.61	0.41	0.43
Many New Ideas	**2.09**	0.08	-0.08	0.05	0.19	0.4	0.06	0.44	0.54	0.42
My New Favorite Author	**4.86**	0.57	0.5	0.4	0.25	0.54	0.64	0.65	0.65	0.66
My New Favorite Story	**4.38**	0.55	0.21	0.27	0.38	0.52	0.61	0.61	0.61	0.62
Page Turner	**2.8**	0.49	0.04	0.09	0.1	0.33	0.49	0.48	0.36	0.43
Review	**4.69**	0.65	0.25	0.45	0.46	0.68	0.61	0.62	0.6	0.37
Satisfying Ending	**3.26**	0.53	0.23	0.34	0.05	0.35	0.22	0.68	0.4	0.45
Something New Under The Sun	**3.86**	0.65	0.4	-0.11	0.44	0.46	0.43	0.58	0.65	0.35
Strange Mood	**-0.02**	0.06	0.27	-0.12	-0.47	0.3	0.34	-0.29	-0.02	-0.08
Taught Me Something	**2.35**	0.24	0.06	-0.06	0.03	0.63	0.33	0.32	0.63	0.17
Too Many Invented Words	**5.38**	0.7	0.14	0.69	0.72	0.61	0.63	0.65	0.68	0.55
Too Many Names	**5.72**	0.73	0.41	0.64	0.46	0.79	0.75	0.53	0.59	0.82
Worth Reading	**4.5**	0.65	0.13	0.31	0.29	0.6	0.6	0.6	0.71	0.62
Would Make A Good Movie	**1.39**	0.54	-0.13	-0.02	-0.03	0.17	0.43	0.07	0.09	0.26
TOTAL	**113.98**	**15.84**	**7.57**	**8.02**	**9.57**	**13.41**	**15.94**	**15.09**	**14.25**	**14.29**

Figure E.2.: Example user correlation matrix: No SciFi fan, with IT background.

u06IS	TOTAL	u01--	u02I-	u03--	u04--	u05--	u07IS	u08IS	u09-S	u10IS
Better Enjoyed With Alcohol	**2.9**	0.52	0.44	0.21	-0.05	0.62	0.31	-0.02	0.5	0.37
Boilerplate Story	**3.75**	0.62	0.24	0.11	0.24	0.42	0.45	0.51	0.56	0.6
Clearly Written	**4.92**	0.46	0.73	0.37	0.43	0.6	0.55	0.71	0.6	0.48
Clever Plot	**3.33**	0.51	0.29	0.44	0.3	0.28	0.34	0.41	0.37	0.4
Convincing Characters	**0.61**	0.24	0.07	0.04	0.1	0.27	-0.01	0.24	-0.06	-0.29
Convincing Plot	**3.06**	0.54	0.34	0.09	0.51	0.2	0.4	0.57	0.04	0.37
Convincing Setting	**4.92**	0.79	0.45	0.31	0.63	0.67	0.51	0.65	0.39	0.52
Convoluted Plot	**2.59**	0.59	0.34	0.2	0	-0.16	0.59	0.66	0.36	0.01
Easy To Read	**4.62**	0.57	0.72	0.35	0.49	0.54	0.38	0.65	0.42	0.49
Entertaining Humor	**6.04**	0.87	0.74	0.57	0.47	0.55	0.69	0.84	0.7	0.61
Good Quotes	**2.25**	0.56	0.19	-0.04	0.22	-0.05	0.37	0.27	0.36	0.35
Good Story Twists	**3.47**	0.34	0.4	0.14	0.35	0.47	0.41	0.47	0.65	0.24
Intriguing Characters	**2.72**	0.47	0.31	0.41	0.17	0.33	0.38	0.27	0.17	0.2
Intriguing Mood	**-0.68**	0.01	-0.06	0.23	-0.38	-0.19	0.23	0.18	-0.28	-0.41
Intriguing Plot	**2.22**	0.31	0.52	-0.16	0.05	0.12	0.4	0.35	0.44	0.18
Intriguing Setting	**3.91**	0.68	0.53	-0.14	0.41	0.66	0.8	0.45	0.16	0.36
Intriguing Writing Style	**-1.9**	-0.46	-0.13	0.02	-0.4	0.06	-0.2	-0.12	-0.2	-0.47
Made Me Angry	**5.47**	0.65	0.38	0.29	0.75	0.57	0.75	0.78	0.72	0.58
Made Me Happy	**3.06**	0.34	0.54	0.23	0.06	0.19	0.52	0.62	0.23	0.33
Many New Ideas	**2.14**	0.14	0.4	-0.12	0.09	0.21	0.32	0.57	0.21	0.32
My New Favorite Author	**5.15**	0.59	0.54	0.61	0.42	0.45	0.38	0.73	0.89	0.53
My New Favorite Story	**5.13**	0.75	0.52	0.42	0.61	0.45	0.58	0.73	0.72	0.35
Page Turner	**1.97**	0.36	0.33	0.04	0.05	0.31	0.33	0.18	0.2	0.16
Review	**4.87**	0.64	0.68	0.27	0.31	0.41	0.82	0.69	0.62	0.43
Satisfying Ending	**2.8**	0.43	0.35	0.16	0.22	0.29	0.14	0.42	0.43	0.37
Something New Under The Sun	**2.99**	0.32	0.46	0.3	0.05	0.32	0.4	0.45	0.34	0.37
Strange Mood	**1.99**	0.42	0.3	0.07	0.18	0.21	0.15	0.09	0.46	0.12
Taught Me Something	**2.1**	0.44	0.63	0.4	-0.29	-0.05	0.49	0.26	0.43	-0.22
Too Many Invented Words	**6.26**	0.84	0.61	0.26	0.66	0.81	0.77	0.96	0.85	0.5
Too Many Names	**5.97**	0.75	0.79	0.48	0.58	0.26	0.87	0.72	0.65	0.87
Worth Reading	**4.92**	0.78	0.6	0.37	0.33	0.4	0.71	0.59	0.63	0.51
Would Make A Good Movie	**1.69**	-0.16	0.17	0.32	0.19	0.43	0.38	0.2	0.22	-0.07
TOTAL	**105.24**	**14.94**	**13.41**	**7.25**	**7.77**	**10.66**	**14.2**	**15.08**	**12.8**	**9.14**

Figure E.3.: Example user correlation matrix: Is a SciFi fan, with IT background.

u09-S	TOTAL	u01--	u02I-	u03--	u04--	u05--	u06IS	u07IS	u08IS	u10IS
Better Enjoyed With Alcohol	**4.5**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Boilerplate Story	**4.23**	0.36	0.51	-0.06	0.34	0.76	0.56	0.51	0.58	0.66
Clearly Written	**4.57**	0.39	0.51	0.21	0.28	0.69	0.6	0.66	0.76	0.47
Clever Plot	**2.65**	0.02	0.51	0.14	0.17	0.52	0.37	0.26	0.34	0.31
Convincing Characters	**4.4**	0.57	0.42	0.52	0.5	0.37	-0.06	0.75	0.52	0.81
Convincing Plot	**1.18**	-0.02	0.39	0.14	0.06	0.06	0.04	0.17	0	0.32
Convincing Setting	**3.38**	0.39	0.47	0.2	0.37	0.25	0.39	0.44	0.42	0.45
Convoluted Plot	**2.42**	0.66	0.08	0.34	-0.18	-0.1	0.36	0.29	0.58	0.39
Easy To Read	**5.41**	0.6	0.53	0.52	0.48	0.7	0.42	0.78	0.87	0.5
Entertaining Humor	**5.37**	0.6	0.55	0.43	0.68	0.72	0.7	0.48	0.65	0.56
Good Quotes	**2.94**	0.35	0.31	0.4	0.62	-0.02	0.36	0.2	0.6	0.11
Good Story Twists	**3.97**	0.43	0.54	-0.03	0.26	0.59	0.65	0.5	0.64	0.39
Intriguing Characters	**2.8**	0.22	0.59	-0.05	0.35	0.27	0.17	0.35	0.42	0.47
Intriguing Mood	**1.68**	0.14	0.52	0.05	0.18	0.45	-0.28	0.17	0.03	0.42
Intriguing Plot	**2.27**	0.17	0.43	0.02	0.19	0.08	0.44	0.12	0.54	0.28
Intriguing Setting	**1.43**	0.33	0.25	-0.23	0.21	-0.13	0.16	0.26	0.15	0.44
Intriguing Writing Style	**1.84**	-0.1	0.21	0.3	-0.2	0.76	-0.2	0.41	0.1	0.57
Made Me Angry	**5.33**	0.78	0.02	0.36	0.71	0.79	0.72	0.67	0.7	0.56
Made Me Happy	**4.58**	0.35	0.41	0.83	0.73	0.67	0.23	0.56	0	0.8
Many New Ideas	**2.58**	0.49	0.54	0.24	0.22	0.19	0.21	0.21	0.24	0.24
My New Favorite Author	**5.64**	0.59	0.65	0.69	0.63	0.4	0.89	0.43	0.71	0.64
My New Favorite Story	**4.87**	0.63	0.61	0.55	0.41	0.46	0.72	0.57	0.62	0.3
Page Turner	**2.59**	0.4	0.36	-0.17	0.35	0.43	0.2	0.32	0.49	0.2
Review	**4.63**	0.65	0.6	0.13	0.45	0.5	0.62	0.64	0.56	0.49
Satisfying Ending	**1.74**	0.13	0.4	-0.06	0.12	0.07	0.43	0.07	0.21	0.37
Something New Under The Sun	**4.41**	0.6	0.65	0.55	0.37	0.45	0.34	0.35	0.64	0.44
Strange Mood	**2.18**	0.39	-0.02	0.11	0.25	0.07	0.46	0.06	0.48	0.38
Taught Me Something	**2.67**	0.44	0.63	0.41	-0.17	-0.11	0.43	0.55	0.33	0.16
Too Many Invented Words	**6.98**	0.95	0.68	0.16	0.74	0.89	0.85	0.94	0.97	0.78
Too Many Names	**5.09**	0.77	0.59	0.3	0.4	0.37	0.65	0.73	0.53	0.76
Worth Reading	**4.86**	0.68	0.71	0.16	0.52	0.5	0.63	0.45	0.58	0.62
Would Make A Good Movie	**1.23**	-0.22	0.09	0.28	0.03	0.41	0.22	0.1	-0.09	0.42
TOTAL	**114.41**	**13.22**	**14.25**	**7.95**	**10.59**	**12.59**	**12.8**	**13.5**	**14.7**	**14.79**

Figure E.4.: Example user correlation matrix: Is a SciFi fan, without IT background.

List of Figures

Index

Bibliography

[ACHZ09] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.

[Ada05] Barbara D. Adams. Trust vs. confidence. Technical report, Defence Research and Development Canada, June 2005.

[Ajz75] I. Ajzen. From intentions to actions: A theory of planned behaviour. In J. Kuhl and J. Beckmann, editors, *Action control: From cognition to behaviour. New York. Springer Verlag*, 1975.

[AL07] Sören Auer and Jens Lehmann. What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In *Proceedings of the ESWC (2007)*, LNCS (4519), pages 503–517. Springer, 2007.

[Ami12] Injy Amin. Personalized views in a distributed annotation system. Master thesis, German University in Cairo, February 2012. Supervised by Malte Kiesel, Slim Abdennadher, Andreas Dengel.

[AT05] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[BC07] Christian Bizer and Richard Cyganiak. D2RQ – Lessons Learned. *W3C Workshop on RDF Access to Relational Databases*, October 2007.

[BCG07] Christian Bizer, Richard Cyganiak, and Tobias Gauss. The RDF Book Mashup: From Web APIs to a Web of Data. In *3rd Workshop on Scripting for the Semantic Web*, June 2007.

[BDF+03] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan. Matching Words and Pictures. *J. Mach. Learn. Res.*, 3:1107–1135, March 2003.

[Beg06] G. Begelman. Automated Tag Clustering: Improving search and exploration in the tag space. In *In Proc. of the Collaborative Web Tagging Workshop*, WWW'06, 2006.

[BEP+08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[BGE+08] Michel Buffa, Fabien Gandon, Guillaume Ereteo, Peter Sander, and Catherine Faron. SweetWiki: A semantic wiki. In *Special Issue of the Journal of Web Semantics on Semantic Web and Web 2.0, Volume 6, Issue 1*, 2008.

[BGGS11] Ansgar Bernardi, Gunnar Aastrand Grimnes, Tudor Groza, and Simon Scerri. The NEPOMUK Semantic Desktop. In Paul Warren, John Davies, and Elena Simperl, editors, *Context and Semantics for Knowledge Management - Technologies for Personal Productivity*, pages 255–274. Springer, September 2011.

[BL+09] Tim Berners-Lee et al. Linked data – the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[BM07] Dan Brickley and Libby Miller. FOAF vocabulary specification. Technical report, FOAF project, May 2007. `http://xmlns.com/foaf/spec/20070524.html`.

[BS97] M. Balabanovic and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, 40:66–72, 1997.

[BSVW12] Francois Bry, Sebastian Schaffert, Denny Vrandecic, and Klara A. Weiand. Semantic wikis: Approaches, applications, and perspectives. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web*, volume 7487 of *Lecture Notes in Computer Science*, pages 329–369. Springer, 2012.

[Bus10] Georg Buscher. *Attention-Based Information Retrieval.* PhD thesis, University of Kaiserslautern, 2010. http://d-nb.info/1007134488.

[BWC07] A. Byde, H. Wan, and S. Cayzer. Personalized Tag Recommendations via Tagging and Content-based Similarity Metrics. In *In Proceedings of the International Conference on Weblogs and Social*, 2007.

[CC06] Ivan Cantador and Pablo Castells. Multilayered semantic social network modelling by ontology-based user profiles clustering: Application to collaborative filtering. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006), Podebrady, Czech Republic. Springer Verlag Lectures Notes in Artificial Intelligence*, pages 334–349. Springer, 2006.

[CCHN07] P. A. Chirita, S. Costache, S. Handschuh, and W. Nejdl. P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web. In *In Proc. of the 16th Intl. World Wide Web Conf*, pages 8–12, 2007.

[CMBT02] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[Con04] World Wide Web Consortium. SWRL: A Semantic Web Rule Language Combining OWL and RuleML., 2004. `http://www.w3.org/Submission/SWRL/`.

[Dav85] F. Davis. A technology acceptance model for empirically testing new end-user information systems: theory and results. *Doctoral dissertation, MIT Sloan School of Management, Cambridge, MA*, 1985.

[Dav93] F. Davis. User acceptance of computer technology: system characteristics, user perceptions. In *Int. J. Man-Machine Studies, 38 (3), 475-487*, 1993.

[Den12] Andreas Dengel, editor. *Semantische Technologien: Grundlagen. Konzepte. Anwendungen.* Spektrum Akademischer Verlag, Heidelberg, 2012. ISBN 978-3827426635.

[DG04] Ido Dagan and Oren Glickman. Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *PASCAL workshop on Learning Methods for Text Understanding and Mining, Grenoble, France*, January 2004.

[DI08] Wisam Dakka and Panagiotis G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE

'08, pages 466–475, Washington, DC, USA, 2008. IEEE Computer Society.

[DJLW08] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.

[Dub11] Dublin Core Metadata Initiative. DCMI Home: Dublin Core Metadata Initiative (DCMI), April 2011.

[Ebe10] Sebastian Ebert. An Application for Fusing Handwriting Recognition and Ontology-Based Information Extraction. Master's thesis, FH Schmalkalden, May 2010.

[FA75] M. Fishbein and I. Ajzen. Belief, Attitude, Intention, and Behaviour: An Introduction to Theory and Research. *Reading, MA; Addison-Wesley*, 1975.

[GA08] D. Godoy and A. Amandi. Hybrid Content and Tag-based Profiles for Recommendation in Collaborative Tagging Systems. In *Proceedings of the 2008 Latin American Web Conference*, LA-WEB '08, pages 58–65, Washington, DC, USA, 2008. IEEE Computer Society.

[Gan07] Aldo Gangemi. DOLCE UltraLite OWL Ontology. `http://www.loa-cnr.it/ontologies/DUL.owl`, 2007.

[GHM+07] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Aastrand Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. The NEPOMUK Project - On the way to the Social Semantic Desktop. In *Proceedings of International Conferences on new Media technology (I-MEDIA-2007) and Semantic Systems (I-SEMANTICS-07), Graz, Austria, September 5-7.*, pages 201–210, 9 2007.

[GHS08] Andreas Gutscher, Jessica Heesen, and Oliver Siemoneit. Possibilities and Limitations of Modeling Trust and Reputation. In Manuel Möller, Thomas Roth-Berghofer, and Wolfgang Neuser, editors, *WSPI*, volume 332 of *CEUR Workshop Proceedings*, pages 50–61, 2008.

[GSS06] Gunnar Grimnes, Sven Schwarz, and Leo Sauermann. RDFHomepage or Finally, a Use For Your FOAF File. In *Proceedings of Semantic*

Web Scripting Workshop at ESWC06, 2006. `http://rdfhomepage.opendfki.de/`.

[Guh96] R.V. Guha. Meta Content Framework: A White Paper. Technical Report 168, Apple, 1996.

[Han05] Siegfried Handschuh. *Creating Ontology-based Metadata by Annotation for the Semantic Web*. PhD thesis, 2005. University of Karlsruhe (TH).

[HC09] Oktie Hassanzadeh and Mariano Consens. Linked Movie Data Base. In *Proceedings of the Linked Data on the Web Workshop (LDOW2009)*, 2009.

[HdOG08] Félix Hernández del Olmo and Elena Gaudioso. Evaluation of Recommender Systems: A New Approach. *Expert Syst. Appl.*, 35(3):790–804, October 2008.

[HJSS06] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Bibsonomy: A social bookmark and publication sharing system. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, pages 87–102, 2006.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[HM06] T. Heath and E. Motta. Personalizing Relevance on the Semantic Web through Trusted Recommendations from a Social Network. In *Proceedings of the Semantic Web Personalization Workshop at the 3rd European Semantic Web Conference*, 2006.

[HM07] Tom Heath and Enrico Motta. Revyu.com: A Reviewing and Rating Site for the Web of Data. *The Semantic Web*, 4825:895–902, 2007.

[HMK07] D. Huynh, S. Mazzocchi, and D. Karger. Piggy Bank: Experience the Semantic Web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):16–27, 2007.

[HRH08] Wolfgang Halb, Yves Raimond, and Michael Hausenblas. Building Linked Data For Both Humans and Machines. In *Proceedings of the Linked Data on the Web Workshop (LDOW2008)*, 2008.

[HSLA09] Sebastian Hellmann, Claus Stadler, Jens Lehmann, and Sören Auer. DBpedia Live Extraction. In *Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, 2009.

[JC09] K. H. Jamieson and J. N. Cappella. *Echo Chamber: Rush Limbaugh and the Conservative Media Establishment*. Oxford University Press, 2009.

[JH09] S. Ju and K. Hwang. A Weighting Scheme for Tag Recommendation in Social Bookmarking Systems. In Folke Eisterlehner, Andreas Hotho, and Robert Jäschke, editors, *ECML PKDD Discovery Challenge 2009 (DC09)*, volume 497, pages 109–118, Bled, Slovenia, September 2009. CEUR Workshop Proceedings.

[JMH+07] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and Gerd Stumme. Tag Recommendations in Folksonomies. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 2007, pages 506–514, Berlin, Heidelberg, 2007. Springer-Verlag.

[KG10] Malte Kiesel and Gunnar Aastrand Grimnes. DBTropes—a linked data wrapper approach incorporating community feedback. In Johanna Völker; Oscar Corcho, editor, *EKAW 2010 Demo and Poster Abstracts. International Conference on Knowledge Engineering and Knowledge Management (EKAW-10), 17th International Conference on Knowledge Engineering and Knowledge Management, October 11-15, Lisbon, Portugal*, October 2010. Best Poster.

[KH13] Malte Kiesel and Injy Hamed. Goal-oriented annotation recommenders. In P. Cimiano, O. Corcho, V. Presutti, L. Hollink, and S. Rudolph, editors, *The Semantic Web: Semantics and Big Data. European Semantic Web Conference (ESWC-13), SWCS2013@ESWC2013 Workshop on Semantic Web Collaborative Spaces, located at ESWC 2013, May 26-30, Montpellier, France*. Springer, 2013.

[Kie06] Malte Kiesel. Kaukolu: Hub of the semantic corporate intranet. In *First Workshop on Semantic Wikis: From Wiki to Semantics co-located with the ESWC-2006*, 2006.

[KM11] Malte Kiesel and Florian Mittag. Personalization in Skipforward, an Ontology-Based Distributed Annotation System. In Marco de Gemmis, Ernesto William De Luca, Tommaso Di Noia,

Aldo Gangemi, Michael Hausenblas, Pasquale Lops, Thomas Lukasiewicz, Till Plumbaum, and Giovanni Semeraro, editors, *SPIM*, volume 781 of *CEUR Workshop Proceedings*, pages 90–97. CEUR-WS.org, 2011.

[KS00] Vipul Y. Kashyap and Amit Sheth. *Information Brokering Across Heterogeneous Digital Data: A Metadata-Based Approach*. Kluwer Academic Publishers, Boston [u.a.], 2000. ISBN 0792378830.

[KSB+10] Thomas Kurz, Sebastian Schaffert, Tobias Buerger, Stephanie Stroka, Rolf Sint, Mihai Radulescu, and Szabolcs Grunwald. KiWi – A Platform for building Semantic Social Media Applications. In *9th International Semantic Web Conference (ISWC2010)*, November 2010.

[KSEB08] Malte Kiesel, Sven Schwarz, Ludger van Elst, and Georg Buscher. Using Attention and Context Information for Annotations in a Semantic Wiki. In C. Lange, S. Schaffert, H. Skaf-Molli, and M. Völkel, editors, *Proceedings of the 3rd Semantic Wiki Workshop (SemWiki 2008) at the 5th European Semantic Web Conference (ESWC 2008), Tenerife, Spain, June 2nd, 2008*, volume 360 of *CEUR-WS*, 2008.

[KSP03] Marja-Riitta Koivunena, Ralph Swick, and Eric Prud'hommeaux. Annotea Shared Bookmarks. In *Proceedings of the KCAP03 workshop*, 2003.

[KSR+09] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In *ESWC 2009 Heraklion: Proceedings of the 6th European Semantic Web Conference on The Semantic Web*, pages 723–737, Berlin, Heidelberg, 2009. Springer-Verlag.

[KSvEB08] Malte Kiesel, Sven Schwarz, Ludger van Elst, and Georg Buscher. Mymory: Enhancing a Semantic Wiki with Context Annotations. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *Proc. of 5th European Semantic Web Conference. The Semantic Web: Research and Applications (ESWC-2008), 5th Conference, June 1-5, Tenerife, Spain*, volume Vol. 5021/2008 of *Lecture Notes in Computer Science, LNCS*, pages 817–821. Springer, 6 2008. Demo Paper.

[KVV05] Markus Krötzsch, Denny Vrandecic, and Max Völkel. Wikipedia and the Semantic Web — The Missing Links. In *Proceedings of Wikimania 2005*. Wikimedia Foundation, JUL 2005.

[KVV06] Markus Krötzsch, Denny Vrandecic, and Max Völkel. Semantic MediaWiki. In I. F. Cruz, St. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942. Springer, 2006.

[LC07] Sigma On Kee Lee and Andy Hon Wai Chun. Automatic Tag Recommendation for the Web 2.0 Blogosphere Using Collaborative Tagging and Hybrid ANN Semantic Structures. In *Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science - Volume 6*, ACOS'07, pages 88–93, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[Lew95] James R. Lewis. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *Int. J. Hum. Comput. Interaction*, 7(1):57–78, 1995.

[LGZ08] X. Li, L. Guo, and Y. E. Zhao. Tag-Based Social Interest Discovery. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 675–684, New York, NY, USA, 2008. ACM.

[LIJ+14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2014.

[LW08] J. Li and J. Z. Wang. Real-Time Computerized Annotation of Pictures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):985–1002, June 2008.

[MA04] P. Massa and P. Avesani. Trust-Aware Collaborative Filtering for Recommender Systems. *Lecture Notes in Computer Science*, pages 492–508, 2004.

[Mar95] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. Dept. of Computing Science and Mathematics, University of Stirling, 1995.

[Mat04] A. Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata. `http://www.adammathes.com/academic/`

computer-mediated-communication/folksonomies.html, 2004.

[Mau96] Ueli Maurer. Modelling a Public-Key Infrastructure. *Lecture Notes in Computer Science*, pages 325–350, 1996.

[MB09] Alistair Miles and Sean Bechhofer. SKOS Simple Knowledge Organization System Reference. W3C Recommendation 18 August 2009, 2009.

[Mic07] E. Michlmayr. Learning User Profiles from Tagging Data and Leveraging them for Personal(ized) Information Access. In *In Proceedings of the Workshop on Tagging and Metadata for Social Information Organization, 16th International World Wide Web Conference (WWW2007)*, 2007.

[Mis06] G. Mishne. AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 953–954, New York, NY, USA, 2006. ACM.

[Mit08] Florian Mittag. Performant Trust and Similarity Metrics for Inconsistent Knowledge-bases. Diplomarbeit, Technische Universität Kaiserslautern, Germany, December 2008. Supervised by Andreas Dengel, Malte Kiesel.

[MKS09] Martin Memmel, Michael Kockler, and Rafael Schirru. Providing Multi Source Tag Recommendations in a Social Resource Sharing Platform. *Journal of Universal Computer Science (JUCS)*, 15(3):678–691, 2009.

[MMB+13] Olena Medelyan, Steve Manion, Jeen Broekstra, Anna Divoli, Anna-Lan Huang, and Ian H. Witten. Constructing a Focused Taxonomy from a Document Collection. In Philipp Cimiano, Oscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *ESWC*, volume 7882 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2013.

[MPPS09] Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C recommendation, W3C, October 2009.

[MS07] Martin Memmel and Rafael Schirru. ALOE – A Socially Aware Learning Resource and Metadata Hub. In Martin Wolpers, Ralf Klamma, and Erik Duval, editors, *EC-TEL (Posters)*, volume 280 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[MSH08] Paul Miller, Rob Styles, and Tom Heath. Open Data Commons, a License for Open Data. April 2008. Copyright is held by the author/owner(s). LDOW2008, April 22, 2008, Beijing, China.

[Nie12] Jakob Nielsen. How Many Test Users in a Usability Study?, 2012. `http://www.nngroup.com/articles/how-many-test-users/`.

[NP01] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In *FOIS Proceedings*, pages 2–9, New York, NY, USA, 2001. ACM.

[Obi] Marek Obitko. Semantic web stack. `http://en.wikipedia.org/wiki/File:Semantic-web-stack.png`.

[ODC+08] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *IJMSO*, 3(1):37–52, 2008.

[OVBD06] Eyal Oren, Max Völkel, John G. Breslin, and Stefan Decker. Semantic Wikis for Personal Knowledge Management. In St. Bressan, J. Küng, and R. Wagner, editors, *DEXA*, volume 4080 of *Lecture Notes in Computer Science*, pages 509–518. Springer, 2006.

[Paz99] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, 1999.

[PBMW99] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[PHS13] Eric Prud'hommeaux, Steve Harris, and Andy Seaborne. SPARQL 1.1 Query Language. Technical report, W3C, 2013. `http://www.w3.org/TR/sparql11-query`.

[Pow07] David M. W. Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

[QHK03] Dennis Quan, David Huynh, and David R. Karger. Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference*, pages 738–753, 2003.

[RBAD10] Thomas Roth-Berghofer, Benjamin Adrian, and Andreas Dengel. Case Acquisition from Text: Ontology-based Information Extraction with SCOOBIE for myCBR. In Isabelle Bichindaritz and Stefania Montani, editors, *Case-Based Reasoning Research and Development: 18th International Conference on Case-Based Reasoning. International Conference on Case-Based Reasoning (ICCBR-2010), 18th, July 19-22, Alessandria, Italy*, volume 6176 of *Lecture Notes in Artificial Intelligence, LNAI*, pages 451–464. Springer Verlag, Heidelberg, 7 2010.

[rdf] RDF primer. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`.

[RHS09] Valentin Robu, Harry Halpin, and Hana Shepherd. Emergence of consensus and shared vocabularies in collaborative tagging systems. *ACM Trans. Web*, 3(4):1–34, 2009.

[Sau09] Leo Sauermann. *The Gnowsis semantic desktop approach to personal information management: weaving the personal semantic web.* PhD thesis, University of Kaiserslautern, 2009. `http://d-nb.info/996092501`.

[SBMD11] Rafael Schirru, Stephan Baumann, Martin Memmel, and Andreas Dengel. Topic-Based Recommendations for Enterprise 2.0 Resource Sharing Platforms. In Andreas König, Andreas Dengel, Knut Hinkelmann, Koichi Kise, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information and Engineering Systems. International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES-2011), 15th, September 12-14, Kaiserslautern, Germany*, volume 6881 of *Lecture Notes in Computer Science, LNAI*, pages 495–504. Springer, 2011.

[Sch10] Sven Schwarz. *Context-Awareness and Context-Sensitive Interfaces for Knowledge Work*. Dissertation, Technische Universität Kaiserslautern, Fachbereich Informatik, March 2010.

[Sch13] Rafael Schirru. *Contextualized Recommendations for the Socio-Semantic Web*. PhD thesis, DFKI GmbH, 2013.

[SDE+06] Leopold Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the EPOS Project. In M. Bouzid and N. Henze, editors, *Proceedings of the International Workshop on Semantic Web Personalization, Budva, Montenegro, June 12, 2006*, pages 42–52, 2006.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[SGW05] Sebastian Schaffert, Andreas Gruber, and Rupert Westenthaler. A Semantic Wiki for Collaborative Knowledge Formation. In *Proceedings of SEMANTICS 2005 Conference*, 2005.

[SK06] Michael Sintek and Malte Kiesel. RDFBroker: A Signature-Based High-Performance RDF Store. In *Proceedings of ESWC 2006*, pages 363–377, Budva, Montenegro, 2006.

[SK11] Sandra Schön and Thomas Kurz. *Smarte Annotationen. Ein Beitrag zur Evaluation von Empfehlungen für Annotationen*. Number 4 in Linked Media Lab Reports. Salzburg Research, Salzburg, 2011.

[SLL+09] X. Si, Z. Liu, P. Li, Q. Jiang, and M. Sun. Content-based and Graph-based Tag Suggestion. In *ECML PKDD DC2009*, pages 243–260, September 2009.

[SM95] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[SOHB07] S. C. Sood, S. H. Owsley, K. J. Hammond, and L. Birnbaum. TagAssist: Automatic Tag Suggestion for Blog Posts. In *In International Conference on Weblogs and Social*, 2007.

[Sou04] Adam Souzis. Rhizome Position Paper, 2004. `http://rx4rdf.liminalzone.org/FOAFPaper`.

[SPMG03] Frank M. Shipman, Morgan Price, Catherine C. Marshall, and Gene Golovchinsky. Identifying Useful Passages in Documents Based on Annotation Patterns. In Traugott Koch and Ingeborg Sølvberg,

editors, *ECDL*, volume 2769 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 2003.

[SRB03] Sven Schwarz and Thomas Roth-Berghofer. Towards goal elicitation by user observation. In *Proceedings of the FGWM 2003 Workshop on Knowledge and Experience Management*, Karlsruhe, 2003.

[SSSS01] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge Processes and Ontologies. *IEEE Intelligent Systems*, 16(1):26–34, January 2001.

[STL11] G. Schroder, M. Thiele, and W. Lehner. Setting Goals and Choosing Metrics for Recommender System Evaluations. In *User-Centric Evaluation of Recommender Systems and Their Interfaces - 2 (UCERSTI 2) workshop, located at 5th ACM Conference on Recommender Systems (RecSys 2011), Chicago, USA*, 2011.

[SvZ08] B. Sigurbjörnsson and R. van Zwol. Flickr Tag Recommendation based on Collective Knowledge. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 327–336, New York, NY, USA, 2008. ACM.

[TFH10] Sebastian Tramp, Philipp Frischmuth, and Norman Heino. OntoWiki – a Semantic Data Wiki Enabling the Collaborative Creation and (Linked Data) Publication of RDF Knowledge Bases. In Oscar Corcho and Johanna Voelker, editors, *Demo Proceedings of the EKAW 2010*, October 2010.

[TKH11] Christoph Trattner, Christian Körner, and Denis Helic. Enhancing the navigability of social tagging systems with tag taxonomies. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, i-KNOW '11, pages 18:1–18:8, New York, NY, USA, 2011. ACM.

[TM07] G. Tummarello and C. Morbidoni. Collaboratively building structured knowledge with DBin: from del.icio.us tags to an RDFS Folksonomy. *Workshop on Social and Collaborative Construction of Structured Knowledge at 16th International World Wide Web Conference (WWW2007)*, 2007.

[UCI+06] Victoria S. Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic

annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4(1):14–28, 2006.

[VBGK09] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 650–665. Springer, 2009.

[vEDAH09] Ludger van Elst, Andreas Dengel, Benjamin Adrian, and Jörn Hees. iDocument: Using Ontologies for Extracting and Annotating Information from Unstructured text. In Bärbel Mersching; Marcus Hund; Zaheer Aziz, editor, *KI 2009: Advances in Artificial Intelligence. Künstliche Intelligenz (KI-2009), September 15-18, Paderborn, Germany*, volume 5803 of *Lecture Notes in Artificial Intelligence, LNAI*, pages 249–256. Springer-Verlag, Heidelberg, 9 2009.

[vEKS+08] Ludger van Elst, Malte Kiesel, Sven Schwarz, Georg Buscher, Andreas Lauer, and Andreas Dengel. Contextualized Knowledge Acquisition in a Personal Semantic Wiki. In Aldo Gangemi and Jérôme Euzenat, editors, *EKAW*, volume 5268 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2008.

[VMDD03] Viswanath Venkatesh, Michael G. Morris, Gordon B. Davis, and Fred D. Davis. User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly*, 27(3):425–478, 2003.

[Wal07] Thomas Vander Wal. Folksonomy, 2007. `http://vanderwal.net/folksonomy.html`.

[WF94] S. Wasserman and K. Faust. Social Network Analysis. Cambridge University Press, Cambridge, 1994.

[XFMS06] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the Semantic Web: Collaborative Tag Suggestions. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.

[ZL04] C. N. Ziegler and G. Lausen. Analyzing Correlation between Trust and User Similarity in Online Communities. *Lecture Notes in Computer Science*, pages 251–265, 2004.

Curriculum Vitae (in German)

Name Malte Kiesel

Berufliche Praxis

2012-2013	Wissenschaftlicher Mitarbeiter am Fachbereich Wissensbasierte Systeme (Prof. Andreas Dengel), Technische Universität Kaiserslautern
2004-2012	Wissenschaftlicher Mitarbeiter bei der DFKI GmbH (Deutsches Forschungszentrum für Künstliche Intelligenz), Forschungsbereich Wissensmanagement (Prof. Andreas Dengel), Kaiserslautern

Ausbildung

1997-2004	Studium der Informatik (Diplom) an der TU Kaiserslautern. Diplomarbeit “Generating and Integrating Evidence for Ontology Mappings” bei der DFKI GmbH.